Juha-Matti Lammi

RFID-tunnisteiden itserekisteröimisjärjestelmä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Automaatiotekniikka

Insinöörityö

22.4.2018

Metropolia

Tekijä Otsikko Sivumäärä Aika	Juha-Matti Lammi RFID-tunnisteiden itserekisteröimisjärjestelmä 29 sivua + 25 liitettä 22.4.2018
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	Automaatiotekniikka
Ammatillinen pääaine	
Ohjaajat	Tietohallintopäällikkö Mikko Mäkelä Lehtori Timo Kasurinen

Opinnäytetyön tavoitteena oli kehittää asiakkaana toimivan Metropolia AMK:n tietohallinnolle toiminnallinen prototyyppi web-sovelluksesta, jota käyttäen opiskelijat ja henkilökunta voisivat rekisteröidä pin-koodeja sekä hallussaan olevia RFID-tunnisteita, kuten HSL-matkakortteja, hakemistopalvelimelle.

Kehitystyö on suoritettu testiympäristössä, johon kuului web-palvelin, hakemistopalvelin, relaatiotietokanta sekä käyttöliittyminä työasemien verkkoselaimia. RFID-tunnisteen lukeminen toteutui työasemaan liitetyltä lukijalaitteelta.

Opinnäytetyö sisältää kuvauksen sovelluksen toiminnasta, kehitysympäristöstä sekä rakenteesta. Liitteissä on esitetty toimivan sovelluksen avoin lähdekoodi.

Toiminnallisen prototyypin tavoitteet täyttyivät suunnitellun mukaisesti.

Avainsanat	Web, Java, Javascript, JSP, Eclipse, LDAP, MySQL, RFID

Author Title Number of Pages Date	Juha-Matti Lammi RFID Tag Self Register System 29 pages + 25 appendices 22 April 2018
Degree	Bachelor of Engineering
Degree Programme	Automation Technology
Professional Major	
Instructors	Mikko Mäkelä, IT-manager Timo Kasurinen, Senior Lecturer

The aim of the study was to develop a functional prototype of a web application for Metropolia UAS's IT administration, which enables students and staff to register pin codes and RFID tags in their possession, such as HSL travel cards, for the directory server.

The development work has been carried out in a test environment that includes a web server, a directory server, a relational database, and client web browsers as user interfaces. The RFID tag was read from a reader attached to the client workstation.

The thesis includes a description of the application's performance, development environment, and structure. The appendices show the open source code for a working application.

Objectives of the functional prototype were fulfilled as planned.

Keywords	Web, Java, Javascript, JSP, Eclipse, LDAP, MySQL, RFID

Sisällys

Lyhenteet ja käsitteet

CSS	*Cascading Style Sheets*. Tekniikka web-sivun visuaalisen ulkomuodo määrittämiseen.
HID	*Human Interface Decives*. Ulkoisen laitteen liityntätapa tietokoneeseen.
HTML	*Hyper Text Markup Language*. Web-sivujen standardi kuvauskieli.
HTTP	*Hypertext Transfer Protocol*. Web-selaimen ja -palvelimen välisen tiedonsiirron protokolla.
HTTPS	*Hypertext Transfer Protocol Secure*. HTTP:n muoto, jossa siirrettävä tieto on suojattu.
Java	Ohjelmointikieli.
Javascript	Ohjelmointikieli.
JSP	*Java Server Page*. Web-sivun rakenteen sisältävä tekstidokumentti.
LDAP	*Lightweight Directory Access Protocol*. Standardoitu verkkoprotokolla.
RFID	*Radio Frequency Identification*. Radiotaajuinen etätunnistusmenetelmä.
SRT	*Self Register Tag*. Opinnäytetyön aiheena olevan sovelluksen projektinimi.
XML	*Extensible Markup Language*. Merkintäkieli.

1 Johdanto

Opinnäytetyön tarkoituksena oli kehittää asiakkaana toimineen Metropolia Ammattikorkeakoulun tietohallinnolle toiminnallinen prototyyppi web-sovelluksesta, jonka avulla opiskelijat sekä henkilökunta pystyisivät itse rekisteröimään ja ylläpitämään soveltuvia RFID-tunnisteita, kuten HSL-matkakortteja, asiakkaan LDAP-hakemistopalvelimelle. LDAP-hakemistopalvelimelta tätä henkilöön sidottua tunnistetietoa voisi mahdollisuuksien mukaan käyttää edelleen erinäisten integraatioiden kautta ulkoisissa järjestelmissä. Tällaisia voisivat olla esimerkiksi kiinteistöjen kulunvalvonta, tulostuspalvelut, kirjastopalvelut ja läppärilainaamo. Integraatioiden toteutus ei kuulunut tämän opinnäytetyön sisältöön.

Opinnäytetyön idea heräsi teknisen vuoropuhelun yhteydessä kesällä 2017, jolloin Metropolian edustajien puolesta tiedusteltiin vastaavanlaisen sovelluksen toimitusmahdollisuuksia. Lyhyen markkinakatsauksen jälkeen ilmeni, että kaupallista toteutusta opinnäytetyön aiheena olevasta sovelluksesta ei löytynyt. Metropolian tietohallintopäällikön ja asiantuntijoiden kanssa käytyjen keskustelujen jälkeen järjestelmän toteutustavaksi muodostui avoimen lähdekoodin sovellus, jonka toteutustapa noudattaisi osittain asiakkaan toiveita, mutta jättäisi myös soveltamisen vapauden.

Sovelluksen toiminnallisen prototyypin toteutus tapahtui testiympäristössä, jossa tämä on kehitetty sekä testattu ja josta tämä on suhteellisen vaivattomasti implementoitavissa LDAP-hakemistopalvelimen sekä Java web-palvelimen omaavaan tuotantoympäristöön. Testiympäristö sisälsi LDAP-palvelimena ja tapahtumalokeja tallentavana MySQL-palvelimena toimivan Raspberry Pin, Apache TomCat-web-palvelimena ja Eclipse-kehitysalustana toimivan Windows 7 -työaseman sekä satunnaisia muita työasemia, joiden Chrome-selaimilta web-sovelluksen toimintaa on testattu.

Ajankäytöllisistä syistä pääpaino toteutuksessa on ollut toiminnallisuudessa eikä niinkään visuaalisessa ilmeessä. Sovelluksen lähdekoodi on vapaa käytettäväksi, mutta on huomioitava, että tämän opinnäytetyön sisältämää koodia ei ole minkään tahon puolesta tietoturva-auditoitu.

2 Web-sovellus

Web-sovelluksella tarkoitetaan tietokoneohjelmaa, joka suoritetaan web-palvelimella ja sen käyttö tapahtuu internetin yli web-selaimella [1].

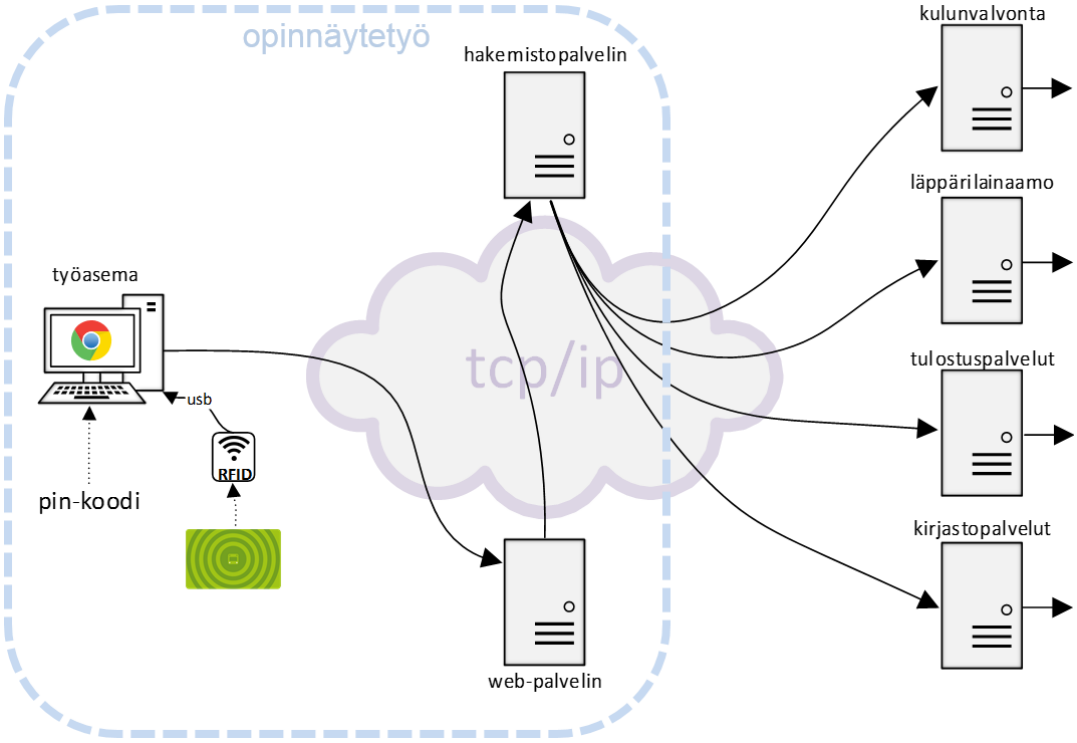
Verkkosivun, eli web-sivun, ja web-sovelluksen ero on lähinnä siinä, että sivu on informatiivinen ja sovellus on interaktiivinen [2]. Esimerkiksi uutissivustoa voitaisiin luonnehtia web-sivuksi ja verkkopankkia web-sovellukseksi. Yleisesti näiden käsitteiden käyttäminen on kuitenkin lähestulkoon mielipidekysymys, ja nykyisin hyvin useat perinteisesti informatiivisena pidetyt sivustot tarjoavat käyttäjälle myös interaktiivista sisältöä [2].

3 SRT-sovellus

Opinnäytetyön alkuvaiheessa kehityskohteena oleva web-sovellus sai projektinimen SRT (Self Register Tag), joka viittaa rfid-tunnisteiden itserekisteröimisominaisuuteen.

3.1 Järjestelmän toimintaperiaate

Järjestelmän toimintaperiaatteena on saattaa järjestelmään kirjatuneen käyttäjän syöttämä rfid-tunniste sekä pin-koodi web-palvelimen välityksellä hakemistopalvelimelle, josta nämä tiedot saatetaan muille järjestelmille erinäisten integraatioiden kautta. Järjestelmään kirjautuminen ja tietojen syöttäminen tapahtuu työaseman selaimella käytettävän SRT-sovelluksen avulla. Kuvassa 1 on havainnollistettu rekisteröitävien tunnistetietojen kulku tietoliikenneverkon yli.



Kuva 1. Järjestelmän toimintaperiaate

Jotta SRT-sovellus toimii, tulee käytettävällä työasemalla olla yhteys web-palvelimeen. Sovellukseen kirjautuminen taas vaatii yhteyden hakemistopalvelimeen, jossa sijaitsevat tiedot käyttäjän todentamiseen.

Käyttäjä tunnistetaan LDAP-hakemistopalvelimelta sisäänkirjautumissivulle asetetun käyttäjätunnuksen sekä salasanan perusteella. Onnistuneen sisäänkirjautumisen yhteydessä käyttäjätunnus ja salasana tallennetaan selainistunnon ajaksi ja näitä käytetään LDAP-yhteyden kirjautumistunnuksina sovelluksen toiminnoissa. Istunnon aikana tehdyt muutokset, esimerkiksi aktiivisen tunnisteen lisääminen, kohdistuvat voimassa olevan istunnon käyttäjätunnukseen.

3.2 SRT-sovelluksen käyttäminen

SRT-sovellusta käytetään verkkoselaimella. Kehitystyön testausvaiheet on suoritettu käyttäen Chrome-selainta, koska asiakkaan määritysten mukaisesti tämä tulisi olemaan ensisijainen selain mahdollisessa loppukäytössä.

Kirjautumissivulla (kuva 2) asetetaan käyttäjätunnus sekä salasana. SRT-sovelluksen kaikki siirtymispainikkeet on ohjelmoitu (ohjlemointikielenä Javascript) siten, että mikäli tarvittavista kentistä ei löydy määrätyn tyyppistä arvoa, on painike harmaa eikä sitä voi painaa. Mikäli tiedot löytyvät, tila vaihtuu aktiiviseksi vihreäksi. Esimerkiksi kirjautumissivulla vaaditaan tieto sekä käyttäjätunnus että salasana -kenttiin, jotta ”Kirjaudu…”-painike aktivoituu. Mikäli käyttäjätunnus on virheellinen tai todennuspalvelimeen (LDAP-palvelin) ei saada yhteyttä, niin sisäänkirjautumissivulle ilmestyy asiaan liittyvä virheviesti. Selaimen osoitekentässä näkyvä ”Ei turvallinen”-teksti viittaa itsekirjoitettuun SSL-sertifikaattiin, jota Chrome-selain ei tunnista.

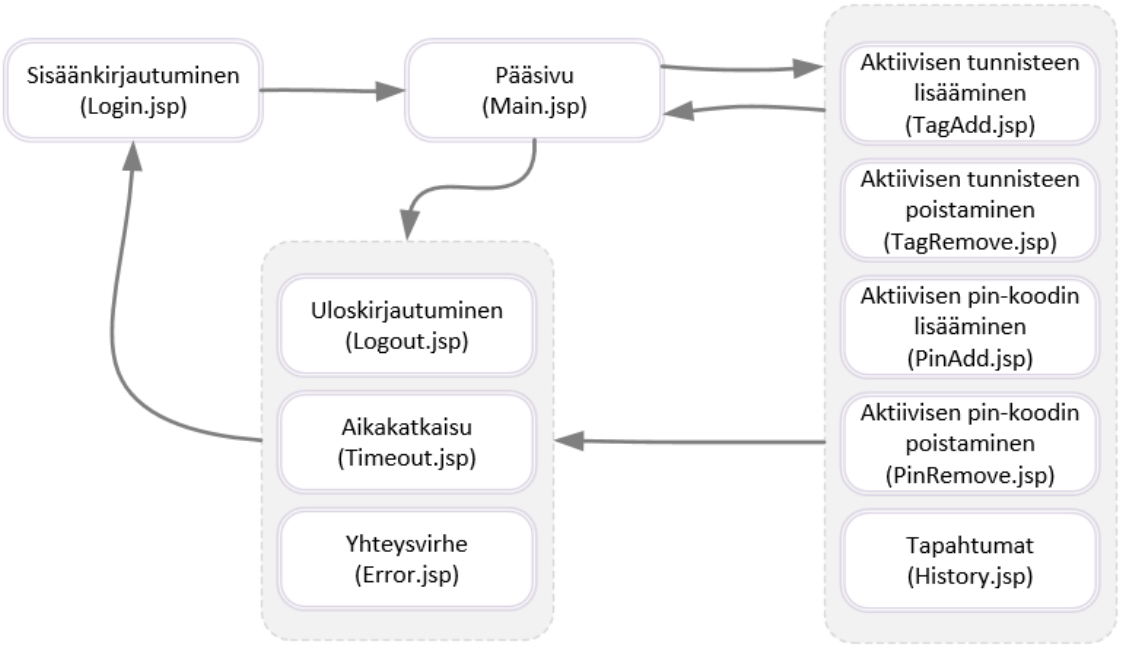
Kuva 2. Sisäänkirjautuminen

Onnistuneen kirjautumisen jälkeen siirrytään pääsivulle (kuva 3). Pääsivun latauksen yhteydessä LDAP-palvelimelta haettujen, käyttäjää koskevien attribuuttien arvot esitetään tekstikentissä, joiden sisältöä ei voi muokata. Tässäkin kaikki painettavat painikkeet ovat vihreinä, riippuen näihin sidottujen tekstikenttien arvoista. Esimerkiksi testiympäristössä käyttäjälle on sallittu vain yhden aktiivisen tunnisteen olemassa olo, joten mikäli hänellä on jo tällainen, niin ainoastaan poisto on mahdollinen. Vastaavasti jos aktiivinen tunniste puuttuu, vain sellaisen asettaminen on mahdollista.

Kuva 3. Pääsivu

SRT-sovelluksen sivujen selainnäkymän kuvakaappaukset löytyvät liitteestä 1.

SRT-sovelluksessa on kaiken kaikkiaan 10 sivua, joiden keskinäinen käyttölogiikka on yksinkertaisuudessaan hyvin samanlainen. Toimintoja sisältäville sivuille siirrytään aina pääsivun kautta, johon myös palataan ennen siirtymistä toiselle sivulle. Pääsivulta sekä kaikilta toimintoja sisältäviltä sivuilta löytyy ”Kirjaudu ulos” -painike, jota painamalla siirrytään uloskirjatuminen-sivulle. Mikäli istunto on päättynyt aikakatkaisun myötä tai jonkin toiminnon yhteydessä ilmenee yhteysvirhe, siirtyy selain automaattisesti näitä vastaaville sivuille. Uloskirjautuminen-, aikakatkaisu ja yhteysvirhe -sivut sisältävät siirtymislinkin sisäänkirjautumissivulle. Siirtyminen sovelluksen sivujen kesken on esitetty kuvassa 4.



Kuva 4. Siirtyminen sovelluksen sivuilla

4 Testiympäristön rakenne

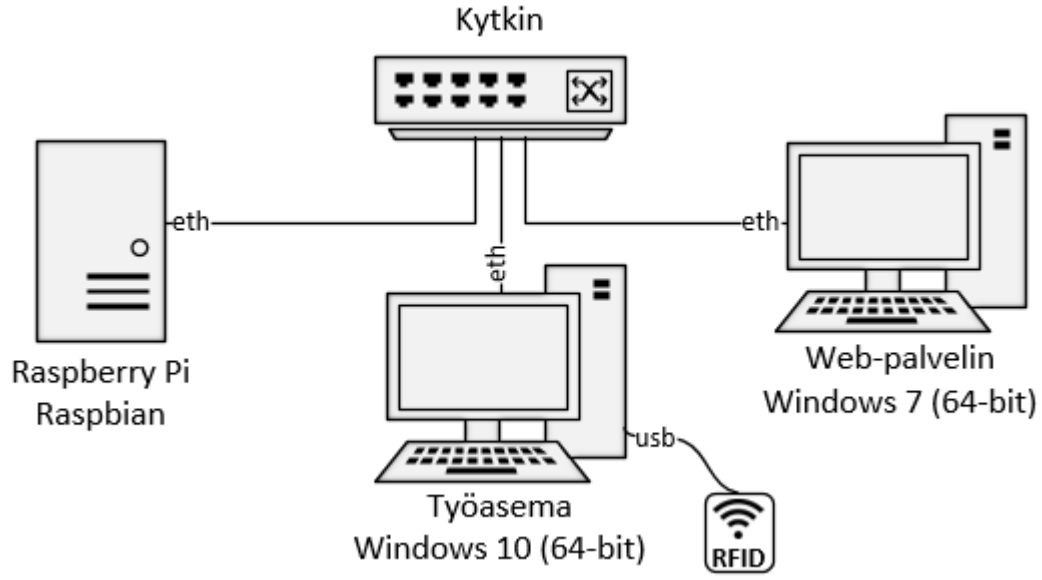
Testiympäristön perustaminen oli ehdoton edellytys, jotta toteutuksen pystyi aloittamaan. Asiakkaan puolelta ehdotettiin mahdollisuutta virtuaaliympäristön käyttämiseksi, mutta joustavuuden vuoksi testiympäristöksi valikoitui repussa kannettava kokoonpano. Tämän myötä toteutuksen eteenpäin vieminen oli mahdollista paikasta riippumatta myös siten, että laiteläheinen RFID-lukijan testaaminen luonnistui vaivatta.

Periaatteessa kaikki sovellukset olisi ollut mahdollista ladata yhdelle Windows-läppärille, johon olisi voitu kytkeä usb-liitäntäinen RFID-lukija ja jonka web-selaimelta olisi myös voitu suorittaa sovelluksen testaaminen. Kuitenkin web- ja hakemistopalvelinsovellusten toiminnallisten erojen hahmottamisen selkeyden vuoksi muodostui päätös eriyttää nämä omille alustoilleen. Vaikkakin pääosa web-sovelluksen testauksesta on suoritettu suoraan web-palvelimen selaimelta, niin työn edetessä ilmeni, että osittain toimintojen testaus on hyvä suorittaa vähintään kahdelta työasemalta yhtäaikaisesti. Tämä auttoi määrittämään http-istunnot toimiviksi silloin, kun sovelluksella on useita yhtäaikaisia käyttäjiä, mikä luonnollisesti on normaali tilanne web-sovellusten käytössä.

Järjestelmän runko toteutui lopulta siten, että Raspberry Pi toimi sekä käyttäjien todentamisen ja tunnistetietojen tallentamisen LDAP-palvelimena että käyttäjälokia tallentavana MySQL-palvelimena. Windows 7 -työaseman rooliksi jäi toimia web-sovelluksen kehitysalustana sekä web-palvelun pyörittäjänä, eli web-palvelimena. Testityöasemalta käytettiin ainoastaan web-selainta, joka kommunikoi https-protokollalla ainoastaan web-palvelun kanssa.

4.1 Hardware ja tietoliikenneyhteydet

Tietoliikenteen osalta testauksen aikana järjestelmää käytettiin pääosin kiinteillä ip-osoitteilla, jolloin Raspberry Pi oli kytkettynä Windows 7 -työasemaan suoralla ethernet-kaapelilla. Web-sovellusta testattaessa ylimääräisen työaseman web-selaimelta, kaikki laitteet olivat kytkettynä paikalliseen tietoliikenneverkkoon, jossa näiden ip-osoitteet tulivat dhcp-palvelulta (kuva 1).



Kuva 5. Hardware ja tietoliikenneyhteydet

RFID-tunnistus

Työasemaan liitetyn USB-liitäntäisen RFID-lukijan lukijan toiminto on lukea käyttäjän tunniste ja asettaa tämän määrätty arvo web-sovelluksen määrättyyn tietokenttään. SRT-sovelluksen osalta määrätty numeerinen arvo on tunnisteen CSN (Card Serial Number), jota joidenkin toimittajien tuotteiden yhteydessä kutsutaan myös UID:ksi (Unique Identifier) [3]. Tunnisteelta luettava ominaisuus määritetään lukijalaitteen ominaisuuksissa.

Opinnäytetyössä käytetty RFID-lukija on ruotsalaisen ASSA Abloyn AB:n tytäryhtiön HID Globalin valmistama OMNIKEY 5427 CK [4]. Asiakkaan määrityksen mukaisesti lukijan tuli olla yhteensopiva Mifare Desfire EV1 -standardin tunnisteiden kanssa, mikä vaikutti mallin valintaan. Kyseisen standardin mukaisia tunnisteita ovat esimerkiksi nykyiset HSL:n matkakortit.

Määrätyn lukustandardin lisäksi lukijalla täytyi olla kyky kirjoittaa merkkejä käyttöliittymään HID-menetelmällä. Perinteisiä HID (Human Interface Devices) -laitteita ovat esimerkiksi tietokoneen näppäimistö sekä hiiri. Näiden tyypillinen käyttötarkoitus on hallita tietokonetta syöttämällä tietoja USB-liitynnän kautta. HID-liityntä mahdollistaa myös kaksisuuntaiseen liikenteen tietokoneen ja laitteen välillä. [5.]

OMNIKEY 5427 CK-lukija ei suoranaisesti tue HID-määritelmää, mutta siinä on Keyboard Wedge -toiminto, jossa laite emuloidaan USB-näppäimistöksi. Lukija siis syöttää saamansa tunnistenumeron järjestelmään, aivan kuin se näppäiltäisiin tietokoneen näppäimistöltä. Tämä toimintamalli on erittäin joustava sovellusten kannalta, koska liityntärajapintaa ei tarvitse erikseen määritellä sovelluksissa. On kuitenkin huomioitava, että laite kirjoittaa saamansa tiedon aina siihen kenttään, missä käyttöjärjestelmän fokus on, kuten normaali näppäimistökin tekisi. [6.]

Vaihtoehtoisesti tunnistetiedon siirtoon oltasiin voitu käyttää lukijalaitetta, joka olisi näkynyt tietokoneen käyttöjärjestelmässä virtuaalisena com-porttina (Virtual COM). Näppäimistöemulaatioon verrattuna tässä olisi tietoturva- ja käytettävyysmielessä sellainen etu, että SRT-sovelluksen tunnistenumeron syöttökenttä olisi voitu lukita, jolloin käyttäjä ei olisi normaalin operoinnin yhteydessä voinut manuaalisesti lisätä tietokoneen näppäimistöltä merkkejä kyseiseen kenttään; ei tietoisesti, eikä vahingossa. Kääntöpuolena tässä menetelmässä olisi se, että koska selaimelta käytettävä web-sovellus ei pääse oletusarvoisesti kiinni tietokoneen com-portteihin, olisi tätä varten jouduttu rakentamaan ylimääräinen liityntärajapinta, joka puolestaan vaatisi oman työasemaa koskevan asennuksen, joka ei ollut asiakkaan näkökulmasta kannatettava vaihtoehto.

Selvyyden vuoksi mainittakoon, että HID Global, jota kutsutaan myös pelkällä HID-nimellä, ja HID (Human Interface Device) ovat kaksi eri asiaa eivätkä varsinaisesti liity toisiinsa.

4.2 Software ja kommunikointiprotokollat

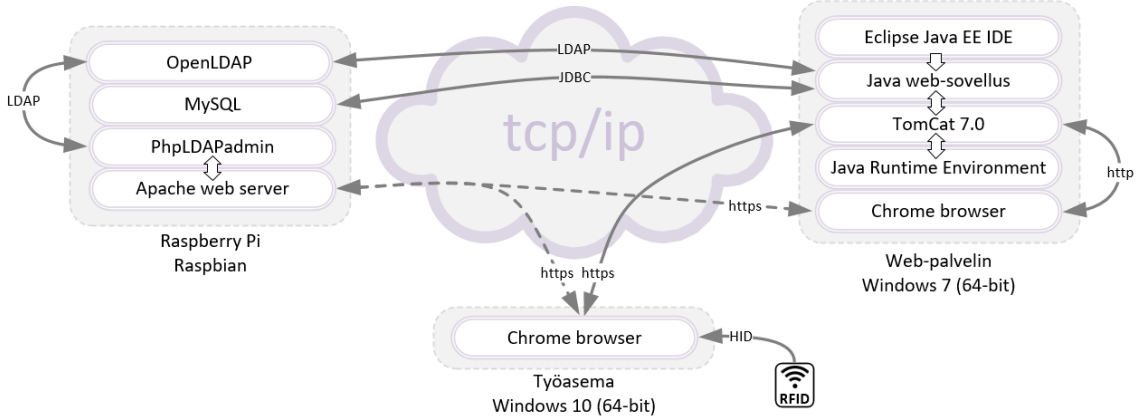
Java-ohjelmointikieleen perustuva selaimelta käytettävä SRT-sovellus toimii web-palvelimen Tomcat-sovelluksessa, jonka palveluun työaseman selain ottaa yhteyden salatun https-protokollan avulla. Sovellus on kehitetty Eclipse-kehitysalustan avulla, josta myös pystytään ohjaamaan Tomcatin toimintoja. Eclipsellä ei kuitenkaan ole välttämättä toiminnallista roolia sovelluksen loppukäytössä. Tomcat vaatii toimiakseen web-palvelimelle asennetun versioyhteensopivan Java-ajoympäristön (Java Runtime Environment, JRE) [7].

Taulukossa 1 on esitetty käytetyt sovellukset ja näiden versiot laitekohtaisesti.

Taulukko 1. Software-versiot

Laite	Sovellus	Versio
Raspberry Pi	OpenLDAP	2.4.45
Raspberry Pi	MySQL	5.7
Raspberry Pi	Apache web server	2.4
Raspberry Pi	PhpLDAPadmin	1.2.2
Web-palvelin	Eclipse Java EE IDE	Kepler Service Release 2
Web-palvelin	Tomcat	7.0
Web-palvelin	Java Runtime Environment	1.8.0_151-b12
Web-palvelin	Chrome-selain	62.0.3202.94
Työasema	Chrome-selain	62.0.3202.94

Kuvassa 6 on havainnollistettu sovellusten keskinäiseen kommunikointiin liittyvät protokollat.



Kuva 6. Kommunikointi

5 SRT-sovelluksen kehitysympäristö

5.1 Eclipse-kehitysalusta

Eclipse on avoimen lähdekoodin kehitysalusta (IDE), joka tarjoaa useita ohjelmointikieliä, viitekehyksiä (framework) ja työkaluja sovellusten kehittämiseen, käyttöönottoon ja ylläpitoon. Alun perin Eclipse-projektin käynnisti IBM vuonna 2001 ja myöhemmin tämän toiminta siirtyi 2004 perustetulle voittoa tavoittelemattomalle Eclipse Foundation -yhteisölle, jonka jäseniä ovat nykyisin muun muassa IBM, Ericsson AB, Oracle, Bosch ja SAP. [8.]

Eclipsestä on useita eri versioita, riippuen kehitettävän sovelluksen ominaisuuksista ja käytettävästä ohjelmointikielestä. SRT-sovelluksen kehitykseen valikoitui ”Eclipse IDE for Java EE Developers”-alusta, joka asentui web-palvelimena ja kehitystyöasemana toimivalle Windows-pc:lle. Jotta Eclipseä voi käyttää, tulee koneelle olla asennettuna JRE (Java Runtime Environment) [9].

5.2 Tomcat

Apache Tomcat, jota yleisimmin kutsutaan nimellä Tomcat, on avoimen lähdekoodin sovelluspalvelinohjelmisto, joka suorittaa Java-koodia sisältäviä Java Servlettejä sekä esittää JSP (Java Server Page) -muotoisia web-sivuja. Tomcat voi toimia itsenäisenä web-palvelimena tai sulautettuna muiden web-palvelinsovellusten, kuten IIS:n (Microsoft Internet Information Server) kanssa. [10.]

Sun Microsystems aloitti Tomcat-projektin referenssisovelluksena Java Servlet ja JSP-teknologioiden esittämiselle. Tomcat-toiminta luovutettiin Apache Software Foundation -yhdistykselle vuonna 1999. [11.]

SRT-sovelluksen Java Servlettien ja JSP-tiedostojen käsittelyä varten web-palvelimena ja kehitystyöasemana toimivalle Windows-pc:lle asentui Tomcat 7.0-versio. Tomcatin pääasiallisen operointi onnistui Eclipse-sovelluksesta erityisen liitännäisen (plugin) avulla. Tomcatin toimintaedellytyksenä on koneelle asennettu JRE (Java Runtime Environment) [7].

5.3 JRE (Java Runtime Environment)

JRE (Java Runtime Environment) on Java-koodiin perustuvien sovellusten ajoympäristö, joka käsittää virtuaalikoneen (JVM - Java Virtual Machine), luokkakirjastot sekä muut tukitiedostot [12]. Eclipsen ja Tomcatin edellyttämänä asensin JRE 1.8.0_151-version web-palvelimena ja kehitystyöasemana toimivalle Windows-pc:lle

5.4 LDAP

LDAP-lyhenne muodostuu sanoista ”Lightweight directory access protocol”. Se on standardoitu verkkoprotokolla, joka käsittää ip-verkon yli tapahtuvan hakemistopalveluiden tietojen käsittelyn asiakasohjelman ja palvelinsovelluksen välillä. [13.]

Hakemistopalvelulla (directory service) tarkoitetaan hakemistopalvelimen (directory server) tarjoamaa palvelua, joka jakaa tietyillä edellytyksillä ip-verkon yli tähän tallennettuja tietoja. Tällaisia ovat esimerkiksi käyttäjän todentamiseen liittyvät tunniste- tai yhteystiedot. [13.]

Sikäli kun hakemistopalvelin käyttää LDAP-protokollaa, niin tätä voidaan yleisesti kutsua myös LDAP-palvelimeksi. LDAP-protokolla ei itsessään ota kantaa siihen, millainen hakemistorakenne tai tietojen tallennustapa LDAP-palvelimella on, vaan se määrittää vain tietyt palvelut tiedonsiirron osalta. [14.]

LDAP-palvelin-sovelluksia löytyy useilta eri toimittajilta. Kaupallisia sovelluksia ovat esimerkiksi Microsoft Active Directory (AD) ja Oracle Internet Directory, kun taas avoimen lähdekoodin sovelluksia ovat esimerkiksi OpenLDAP ja 389 Directory Server. [15.]

Microsoft Active Directoryn lyhenteen ”AD” osalta mainittakoon, että yleisesti tätä käytetään hieman virheellisesti yleisnimityksenä hakemistopalveluille, vaikkakin se on ainoastaan yhden toimittajan vaihtoehto toteuttaa kyseinen palvelu. AD voi toimia LDAP-palvelimena, mutta se ei ole ainoastaan LDAP-protokollasta riippuvainen sovellus, vaan tarjoaa myös muita yhteysrajapintoja. [14.]

OpenLDAP

Opinnäytetyössä käytetyn hakemistopalvelun alustaksi valikoitui Raspberry Pi:lle asennettu OpenLDAP-palvelinsovellus. OpenLDAP on avoimen lähdekoodin hakemistopalvelinsovellus, jonka toiminta perustuu täysin LDAP-protokollaan [16]. Kurt Zeilenga aloitti OpenLDAP-projektin 1998 ja tällä hetkellä sitä ylläpitää ja koordinoi Yhdysvalloissa sijaitseva voittoa tavoittelematon yhdistys OpenLDAP Foundation [17].

SRT-sovelluksen tarkoitus on lisätä LDAP-palvelimelle määrätyt käyttäjän tiedot, joita ovat

- aktiivisen tunnisteen numero
- aktiivisen tunnisteen kuvaus
- aktiivinen pin-koodi.

Testiympäristössä edellä mainittuja käyttäjän tietoja vastaavat OpenLDAP-palvelimen oletusattribuutit ovat

- TelephoneNumber
- Title
- TelexNumber.

Esimerkiksi kun käyttäjä lisää aktiivisen pin-koodin, niin OpenLDAP:ssa hänen käyttäjätunnuksensa alle lisätään attribuutti telexNumber, joka saa arvokseen lisätyn nelinumeroisen pin-koodin. Vastaavasti kun käyttäjä poistaa aktiivisen pin-koodin, telexNumber-attribuutti poistetaan. Oletusarvoisesti LDAP-attribuuttiin ei sallita tyhjää arvoa [18].

Alun perin suunnittelmana oli, että LDAP-palvelimelle olisi lisätty SRT-sovelluksen käsittelemiä tietoja varten kustomoidut attribuutit, kuten tagNumber, tagDescription ja pinCode. Vaikkakin tämä olisi ollut periaatteessa mahdollista, niin asiakkaan asiantuntijoiden lausunnot vaikuttivat ajatuksesta luopumiseen, koska loppukäytössä tultaisiin luul-

tavimmin käyttämään vain vakioattribuutteja. Pintapuolinen perehtyminen LDAP-attribuuttien kustomointiin osoitti tämän olevan ennakko-oletusta huomattavasti monimutkaisempi toteutettava, mikä vaikutti myös ajankäytöllisistä syistä pitäytymiseen OpenLDAP:in oletussisällössä.

5.5 MySQL

MySQL on avoimen lähdekoodin relaatiotietokantajärjestelmä, joka skaalautuu datan säilytyksen ja käytettävyyden osalta sekä pieniin paikallisiin työasemasovelluksiin että suuriin usean palvelimen klusteriratkaisuihin. Tietokantana sitä pidetään erityisen soveltuvana internetin hakupalveluissa, koska se on nopea ja luotettava. Alun perin tuotteen kehitti 1995 ruotsalainen MySQL AB -yritys, mutta sittemmin se siirtyi Oracle Corporation -yrityksen omistukseen. [19.]

SRT-sovelluksessa MySQL-tietokanta toimii kirjautuneen käyttäjän suorittamien tapahtumien, kuten aktiivisen tunnisteen lisäämisen tai poistamisen, tallennuspaikkana. Vastaavasti tallennettuja tapahtumia haetaan SRT-sovelluksen Tapahtumat-osiosta. Tietokanta ”srtdb” sisältää yhden taulun ”SrtUserLog”, josta löytyy kolumnit

- ID (BIG INT, PRIMARY KEY AUTO_INCREMENT)
- Username (VARCHAR(12))
- Timestamp (DATETIME, DEFAUL CURRENT_TIMESTAMP)
- Event (VARCHAR(100)).

SRT-sovelluksen osalta asiakkaalla ei ollut erityistä vaatimusta tapahtumien tallennusmenetelmästä ja MySQL-liityntä onkin toteutettu lähinnä testimielessä. Joka tapauksessa lokitietojen tallentaminen on erittäin olennainen osa järjestelmän toimintaa ja se tulisi sovelluksen osalta jopa ehdollistaa siten, että ilman onnistunutta lokikirjausta ei myöskään tapahdu toiminnallisia muutoksia. Lokitietojen tallentaminen tullaankin epäilemättä toteuttamaan mahdollisessa tuotantoversiossa jollain asiakkaan tietoliikenne- ja datantallennusinfraan soveltuvalla tavalla.

5.6 PhpLDAPadmin-sovellus

PhpLDAPadmin-sovellus on kevyt LDAP-palvelimen ylläpidon työkalu, jonka avulla voidaan toteuttaa muun muassa organisaatioiden ja käyttäjien hallinta selaimen graafisen käyttöliittymän kautta [20].

Vaikkakin Raspberry Pi:lle asennetun OpenLDAP:in hallinta olisi ollut kaikin puolin mahdollista hoitaa konsolista tekstikomennoin, niin tämä osoittautui hieman hankalaksi tilanteessa, jossa SRT-sovelluksen kehitystyön osalta käyttäjien attribuutteihin tuli jatkuvia muutoksia, joiden toteuma tuli tarkastaa. PhpLDAPadmin palveli tätä käyttötarkoitusta mainiosti, vaikkakin se on ominaisuuksiltaan suhteellisen rajattu.

PhpLDAPadminin asennus konfiguroi automaattisesti Raspberry Pi:n Apache web-palvelun, jonka kautta selainkäyttöliittymä toimii.

6 SRT-sovelluksen rakenne

Toiminnalliset komponentit toteutuneessa SRT-sovelluksessa ovat Java Servletit ja muut Java-luokat (Class, Listener, Filter), JSP:t, CSS-tiedosto sekä web-xml -määritystiedosto. Näiden rakentaminen ja testaus on suoritettu yksinomaan Eclipse-sovelluksella. MySQL-tietokantayhteys vaati mysql-connector -kirjaston lisäämisen, mutta muilta osin JRE:n ja Tomcatin asennusten myötä saadut oletuskirjastot riittivät sovelluksen toimintakuntoon saattamiseksi.

Eclipsen Project Explorer -näkymä SRT-sovelluksesta on esitetty liitteessä 2.

6.1 Servlet

Servlet on Java-ohjelmointikielen luokka, joka mahdollistaa palvelinsovelluksen kommunikoinnin vastaus-pyyntö (*request-response*) -mallin mukaisesti, eli kuten web-sivut ja -sovellukset toimivat. Perus Java-luokan laajan kirjastovalikoiman lisäksi Servlet tarjoaa muun muassa http-palvelun mukaiset pyyntömetodit GET ja POST, jotka Servleteissä oletusarvoisesti nimetään doGet ja doPost. [21.]

Esimerkki: käyttäjän selaimella painama painike lähettää POST-pyynnön palvelimen sovelluksen Servletille, jossa ajetaan doPost-metodin sisältämä Java-koodi, jossa muiden toimintojen ohella määritetään selaimelle lähetettävä vastaus.

SRT-sovelluksessa on luotu oma Servlet-luokka jokaiselle sivulle, joka sisältää jotain palvelimen päässä suoritettavaa logiikkaa,

- Login.java
- Main.java
- PinAdd.java
- PinRemove.java
- TagAdd.java

- TagRemove.java

- History.java.

Edellä mainittujen doPost-metodeissa on käsitelty kutakin näitä vastaavan web-sovelluksen sivun POST-pyyntö. Varsinainen palvelinpään toiminta, kuten yhteys LDAP- tai MySQL-palvelimeen sekä näiden kanssa suoritettava tietojen vaihtaminen on käsitelty perus Java-luokkien CommonLDAP.java ja CommonMySQL.java metodeissa, joita käytetään Servlet-luokissa.

SRT-sovelluksen Servlettien sisältämät koodit löytyvät liitteistä 3–9.

6.2 HttpSessionListener

HttpSessionListener-luokkaa käytetään web-sivujen tai -sovellusten http-istuntojen hallintaan. Luokan sessionCreated ja sessionDestroyed -metodien avulla voidaan saavuttaa näiden sisältämän koodin toiminnallisuudet, kun esimerkiksi sovellukseen kirjaudutaan sisään (sessionCreated) ja tästä kirjaudutaan ulos (sessionDestroyed), minkä myötä aktiivisten käyttäjien määrää pystytään valvomaan ja mahdollisesti keräämään näistä historiatietoja. [24.]

SRT-sovelluksessa on yksi HttpSessionLister-luokka

- CommonHttpSessionListener.java.

Tässä luokassa sessionCreated ja sessionDestroyed -metodit yksinkertaisesti kirjoittavat viestejä web-palvelimen konsolinäyttöön, kun istunto alkaa tai päättyy. Istunto voi päättyä uloskirjautumisella tai web-xml-määritystiedostossa asetetun timeout-ajan täytyttyä. Sovelluksen toiminnan kannalta tällä ei siis ole varsinaista merkitystä, vaan pääasiallinen käyttötarkoitus on ollut toimia testaustyökaluna kehitystyössä. Mikäli sovellus otetaan laajempaan käyttöön, niin tätä ominaisuutta kannattaa ehdottomasti käyttää jo ylläpidolle saatavan statistiikan kannalta.

SRT-sovelluksen HttpSessionListenerin sisältämä koodi löytyy liitteestä 10.

6.3 Filter

Filter-luokassa määritetään toiminnot, jotka suoritetaan ennen määrättyyn Servlettiin kohdistuvaan pyyntöön vastaamista. Tätä käytetään usein esimerkiksi lokitietojen tuottamiseen, pyynnön todentamiseen määrätyille resursseille, tietojen formatointiin ennen vastausta ja evästetietojen hallintaan. Web-xml-määritystiedostossa asetetaan, että millä web-sovelluksen sivuilla kyseinen Filter on käytössä. [25.]

SRT-sovelluksessa on yksi Filter-luokka

- CommonHttpCacheFilter.java.

Tässä luokassa on määritetty doFilter-metodin kautta toiminto, joka istunnon päättymisen yhteydessä poistaa selaimen välimuistiin kertyneet tiedot. Näin ollen esimerkiksi selaimen paluupainikkeella palattaessa edeltäville sivuille, näiden aiemmin sisältämät tiedot eivät ole enää käytettävissä. HttpCacheFilter on käytössä kaikilla SRT-sovelluksen sivuilla.

SRT-sovelluksen Filterin sisältämä koodi löytyy liitteestä 11.

6.4 Java-luokka: CommonLDAP.java

SRT-sovelluksen CommonLDAP-luokka sisältää metodit

- LdapLoginSearchAttributes
- LdapSearchAttributes
- LdapPinAdd
- LdapPinRemove
- LdapTagAdd
- LdapTagRemove.

LdapLoginSearchAttributes-metodia käytetään SRT-Login-Servletin doPost-metodin sisällä, kun käyttäjä kirjautuu sovellukseen painamalla ”Kirjaudu…”-painiketta. Tässä suoritetaan käyttäjän todentaminen LDAP-palvelimelta http-istuntoon tallennettujen Login-sivulla syötettyjen kirjautumistietojen avulla, minkä jälkeen kyseisen käyttäjän määrätyt tiedot haetaan LDAP-attribuuteista ja asetetaan http-istunnon attribuuteiksi. Onnistuneen toiminnon myötä siirrytään Main-sivulle, jossa esitetään käyttäjän tiedot. Virhetilanteessa palataan Login-sivulle, johon asetetaan virheviesti: ”Virheellinen käyttäjätunnus tai salasana” tai ”Ei yhteyttä autentikointipalveluun. Yritä hetken kuluttua uudelleen.”.

LdapLoginSearch-metodia käytetään PinAdd, PinRemove, TagAdd ja TagRemove -Servlettien doPost-metodien sisällä, kun käyttäjä painaa kyseisillä sivuilla ”KYLLÄ” tai ”PERUUTA”-painiketta. Tässä suoritetaan käyttäjän todentaminen LDAP-palvelimelta http-istuntoon tallennettujen Login-sivulla syötettyjen kirjautumistietojen avulla, minkä jälkeen kyseisen käyttäjän määrätyt tiedot haetaan LDAP-attribuuteista ja asetetaan http-istunnon attribuuteiksi - sikäli kun nämä ovat jo olemassa sisäänkirjautumisen myötä, niin attribuuttien arvot vain päivitetään. Onnistunut toiminto siirtää käyttäjän takaisin Main-sivulle. Virhetilanteessa, jonka voi aiheuttaa esimerkiksi http-istunnon käyttäjän epäonnistunut todentaminen LDAP-palvelimelta tai LDAP-palvelimen yhteyden aikakatko (timeout), siirrytään Error-sivulle ja http-istunto päätetään.

LdapPinAdd-metodia käytetään PinAdd -Servletin doPost-metodissa. LdapPinAdd määrittää käyttäjän aktiivisen pin-koodin asettamisen attribuutiksi LDAP-palvelimelle, kun painetaan PinAdd-sivun ”KYLLÄ”-painiketta, joka aktivoituu vasta kun oikean muotoiset pin-koodit ovat syöttökentissä. Onnistuneen LDAP-lisäyksen jälkeen ajetaan Common-MySQL-luokan metodi MySQLInsert, joka lisää käyttäjälokiin tiedon aktiivisen pin-koodin asettamisesta. Vaikkakin PinAdd-sivun Javascriptissä käsitellään pin-koodin oikeellisuus, joka on neljä numeroa, niin sama tarkistus suoritetaan myös LdapPinAdd-metodissa. Jos arvo on virheellinen, niin lisäystä ei suoriteta LDAP-palvelimelle eikä käyttäjälokiin.

LdapPinRemove-metodia käytetään PinRemove -Servletin doPost-metodissa. LdapPinRemove poistaa käyttäjän aktiivisen pin-koodin LDAP-attribuutin, kun painetaan PinRemove-sivun ”KYLLÄ”-painiketta. Onnistuneen LDAP-poiston jälkeen ajetaan Common-MySQL-luokan metodi MySQLInsert, joka lisää käyttäjälokiin tiedon aktiivisen pin-koodin poistamisesta.

LdapTagAdd-metodia käytetään TagAdd -Servletin doPost-metodissa. LdapTagAdd määrittää käyttäjän aktiivisen tunnisteen asettamisen attribuutiksi LDAP-palvelimelle, kun painetaan TagAdd-sivun ”KYLLÄ”-painiketta, joka aktivoituu vasta kun oikeanmuotoinen tunnistenumero ja tunnisteen kuvaus ovat syöttökentissä. Onnistuneen LDAP-lisäyksen jälkeen ajetaan CommonMySQL-luokan metodi MySQLInsert, joka lisää käyttäjälokiin tiedon aktiivisen tunnisteen asettamisesta. Vaikkakin SRT_TagAdd-sivun Javascriptissä käsitellään tunnisteen numeron ja kuvauksen oikeellisuus, niin sama tarkistus suoritetaan myös LdapTagAdd-metodissa. Jos arvo on virheellinen, niin lisäystä ei suoriteta LDAP-palvelimelle eikä käyttäjälokiin. Lisäksi epäonnistuneeseen lisäykseen johtaa LdapTagAdd-metodissa suoritettava tarkastus, joka varmistaa, ettei LDAP:sta löydy vastaavaa tunnistenumeroa, jonka tulee olla uniikki.

LdapTagRemove-metodia käytetään TagRemove-Servletin doPost-metodissa. LdapTagRemove poistaa käyttäjän aktiivisen tunnisteen numeron ja kuvauksen LDAP-attribuutit, kun painetaan TagRemove-sivun ”KYLLÄ”-painiketta. Onnistuneen LDAP-poiston jälkeen ajetaan CommonMySQL-luokan metodi MySQLInsert, joka lisää käyttäjälokiin tiedon aktiivisen tunnisteen poistamisesta.

SRT-sovelluksen CommonLDAP-luokan sisältämä koodi löytyy liitteestä 12.

6.5 Java-luokka: CommonMySQL.java

SRT-sovelluksen CommonMySQL-luokka sisältää metodit

- MySQLInsert
- MySQLSelect.

MySQLInsert-metodia käytetään CommonLDAP-luokan LdapPinAdd-, LdapPinRemove-, LdapTagAdd- ja LdapTagRemove-metodeissa, kun tietokannan käyttäjälokitauluun lisätään tapahtumarivi. Ajettaessa MySQLInsert-metodi, tämä saa toimenpiteen kohteena olevan käyttäjän käyttäjätunnuksen (username) sekä kuvauksen (sqlLogEvent) suoritetusta toimenpiteestä, jotka lisätään omina arvoinaan käyttäjälokitaulun uudelle tapahtumariville.

MySQLSelect-metodia käytetään Main-servletin doPost-metodissa, kun Main-sivulla painetaan ”Tapahtumat”-painiketta, jonka myötä siirrytään History-sivulle. MySQLSelect-metodissa haetaan käyttäjälokitaulusta kirjautuneen käyttäjätunnuksen perusteella kaksikymmentä viimeisintä tapahtumaa. Nämä tallennetaan sovelluksen events-attribuuttiin, joka esitetään History-sivulla. Mikäli MySQL-haussa tapahtuu virhe, esimerkiksi tietokantaan ei saada yhteyttä, niin events saa arvon ”Tapahtumahistoria ei ole tällä hetkellä saatavilla”.

SRT-sovelluksen CommonMySQL-luokan sisältämä koodi löytyy liitteestä 13.

6.6 JSP

JSP (Java Server Page) on web-sivun rakenteen sisältävä tekstidokumentti, joka tukee perinteisten tekstipohjaisten formaattien, kuten HTML ja XML, lisäksi dynaamista sisältöä. JSP:n osalta dynaaminen sisältö voi tarkoittaa esimerkiksi HTML-tekstiin lisättyä Java-koodia, joka suoritetaan käyttäjän selaimen sijaan web-palvelimella, tai EL:n (Expression Language) avulla web-sivulle haettuja palvelinpään Java-objektien sisältöjä, kuten muuttujia tai funktioita. JSP-perustaiset web-sivut vaativat toimiakseen Java-pohjaisen web-palvelimen, kuten Tomcatin. [26.]

SRT-sovelluksen kaikki sivut ovat JSP-tyyppisiä. Varsinaista Java-koodia ei ole kuitenkaan sivurakenteisiin lisättyinä, vaan toiminnallinen Java-koodi on sijoitettuina Servletteihin ja muihin Java-luokkiin. Sivuilla ilmenevien, LDAP- ja MySQL-palvelimilta haettavien muuttujien arvojen esitys on kuitenkin toteutettu käyttäen sivun html-koodissa JSP-sivuille ominaista tyyliä: ${muuttuja}. Eli esimerkiksi sivun tekstilaatikkoon asetetaan arvo ${firstname}, joka taas määritetään Java-koodissa funktiolla setAttribute(”firstname”, firstname), joista jälkimmäinen muuttuja on LDAP-palvelimelta haettu käyttäjän etunimi. Joillain sivuilla, kuten Logout.jsp, ei käsitellä lainkaan muuttajatietoa ja ne voivat olla muutenkin toiminnaltaan lähinnä informatiivisia. Tämän vuoksi sivu voisi olla perinteiseen tapaan pelkästään HTML-tyyppinen, mutta yhtenäisyyden sekä selkeyden vuoksi kaikista sovelluksen sivuista on tehty JSP-tyyppisiä.

SRT-sovelluksen JSP-sivujen sisältämät koodit löytyvät liitteistä 14–23.

6.7 HTML5

HTML (Hyper Text Markup Language) on standardi kuvauskieli web-sivujen ja -sovellusten kehittämisessä. HTML-tiedoston teksti rakentuu muun muassa HTML-tageista ja näiden esittelemistä HTML-elementeistä. Tiedoston rakenteen perusteella web-selain tulkitsee käyttäjälle näkyvän web-sivun sisällön. [27.]

HTML-standardin ajankohtaisin versio on HTML5, jota on myös käytetty SRT-sovelluksen JSP-sivuilla.

6.8 CSS

CSS (Cascading Style Sheets) on tekniikka, jonka avulla määritetään web-sivun elementtien, kuten tekstikenttien tai painikkeiden, visuaalinen ulkomuoto, joka käsittää määritykset muun muassa muodolle, sijoittelulle, väritykselle ja käytettävälle fontille. Yleisimmin CSS:ää käytetään ulkoisesta CSS-tiedostosta, joka linkitetään web-sivulle HTML-koodin määrityksissä. Tiedoston käyttämisen merkittävänä etuna on se, että samaa tiedostoa ja tähän määritettyjä tyylisääntöjä voidaan käyttää useilla sivuilla tai sivustoilla. Vaihtoehtoisena menetelminä voidaan käyttää elementti- tai sivukohtaista tyylimäärittelyä. [28.]

SRT-sovelluksessa on käytetty paikallista itse rakennettua CSS-tiedostoa CommonCSS.css, joka on linkitettynä jokaisen JSP-sivun HTML-koodissa. Tyylilliset määritykset tässä ovat hyvin alkeellisella tasolla. Määrityksissä on otettu kantaa lähinnä elementtien horisontaaliin keskitykseen, kokoon sekä väritykseen.

SRT-sovelluksessa käytetyn CommonCSS.css-tiedoston sisältämä koodi löytyy liitteestä 24.

6.9 Javascript

Javascript on ohjelmointikieli, jota käytetään yleisesti web-sivujen kevyen dynaamisen sisällön tuottamiseen. Javascript-koodi voi olla sisällytettynä sivun HTML-koodiin, jolloin

tämä suoritetaan käyttäjän selaimessa. Merkittävinä etuina tässä onkin palvelinpään resurssien keventäminen sekä käyttäjälle näkyvän, välittömästi selaimessa käsiteltävän, palautteen saaminen. [29.]

Esimerkiksi web-sivulla voi esiintyä tekstikenttä, johon käyttäjän tulee kirjoittaa sähköpostiosoitteensa ja painaa lähetä-painiketta, joka ei ole oletusarvoisesti aktiivinen. Javascriptillä voidaan tehdä selaimen päässä suoritettava tarkastus sähköpostiosoitteen oikeasta muodosta, kuten @-merkin sisältymisestä, minkä toteuduttua lähetä-painike aktivoidaan.

SRT-sovelluksessa Javascriptiä on käytetty pääasiassa juurikin sivujen painikkeiden aktiivisuus-tilan vaihtamiseen, johon voi sivun latautumisvaiheessa vaikuttaa Servletin LDAP-palvelimelta saamien attribuuttien arvot tai sivulla esiintyvien syöttökenttien oikeelliset arvot.

SRT-sovelluksessa käytetyt Javascript-koodit löytyvät JSP-sivujen liitteistä 14–23.

6.10 Web.xml-määritystiedosto

Web.xml-määritystiedosto on web-sovelluksen asettelun deskriptori, eli eräänlainen kuvaaja, joka sisältää käytettävien komponenttien määritykset [30]. Komponentteja voivat olla esimerkiksi servletit, filtterit tai listenerit, joista SRT-sovelluksen web.xml-tiedostokin on rakentunut.

SRT-sovelluksessa käytetty web.xml-tiedosto löytyy liitteestä 25.

7 Tietoturvallisuus

Kokonaisuudessaan SRT-sovelluksen toiminnassa on useita tietoturva-aspekteja, joita on havainnoitu osittain kokemukseen, sovellustestaukseen, tietoturva-artikkeleihin sekä asiakaspalautteeseen perustuen. Sovelluksen kehityksen ja testauksen yhteydessä on huomioitu

- tietoliikenteen salaus
- selaimen välimuistin tyhjennys istunnon päätyttyä
- selaimen istunnon aikakatkaisu
- sovelluksen syöttökenttien historiatietojen automaattinen tyhjennys
- lisättävän tunnistenumeron LDAP-attribuutin yksilöllisyys
- pin-koodin käsittely selaimen tietokentissä
- SQL-injektio.

Tietoliikenteen salaamiseksi SRT-sovelluksen käyttöselain keskustelee web-palvelimen kanssa https-protokollalla. Testiympäristön web-palvelimelle on asennettu itse allekirjoitettu SSL-sertifikaatti, joka leimataan oletusarvoisesti esimerkiksi Chrome-selaimessa ei-luotetuksi. Loppukäytössä asiakas tulisi luonnollisesti käyttämään omia sertifikaatteja tietoliikenteen salauksen osalla.

SRT-sovelluksen JSP-sivujen tietoturvaa on parannettu oletusarvoista niiltä osin, että istunto voi päättyä uloskirjautumisen ohella web-xml-tiedostossa määritellyn aikakatkaisun (timeout) puitteissa. Kun istunto päättyy, niin CacheFilter hoitaa selaimen välimuistin tyhjentämisen siten, että esimerkiksi paluupainikkeella ei näe uloskirjautuneen käyttäjän tietoja. Lisäksi kaikkien sovelluksen sivujen syöttökentistä on html-koodissa poistettu autocomplete-ominaisuus, joka muistaisi ja ehdottaisi edellisiä syöttöjä annettaviksi arvoiksi.

Käyttäjän LDAP-attribuuttien osalta aktiivisen tunnistenumeron tulee olla ehdottomasti yksilöllinen tai muuten integroitava järjestelmä ei pysty kohdistamaan tunnistetietoa yksittäiseen henkilöön, mikä on luotettavan tunnistuksen edellytys. Asiakkaan pyynnöstä toteutin tunnistenumeron yksilöllisyyden tarkastuksen SRT-sovelluksen päässä siten, että kun käyttäjä lisää uuttaa aktiivista tunnistetta, niin sovellus tekee systeemikäyttäjänä haun LDAP-hakemistopalvelimeen. Mikäli haku löytää vastaavan tunnistenumeron, niin lisäys epäonnistuu ja sovellus palaa pääsivulle. Mikäli haku ei löydä vastaavuutta, niin aktiivisen tunnisteen lisäys onnistuu.

SRT-sovelluksen syöttö- ja tekstikentissä ilmenevän nelinumeroisen pin-koodin numerot esitetään kaikissa tilanteissa tähti-merkkeinä, vaikkakin tämän esitystapa LDAP-palvelimen attribuutissa on neljä numeroa.

MySQL-yhteyden insert- ja select-lauseissa on käytetty prepared statement -menetelmää, joka osaltaan ehkäisee SQL-injektiota.

Edellä mainittujen menetelmien lisäksi tiedostettuja merkittäviä tietoturvahaasteita ja tähän liittyvää asiakkaan jatkosuunnittelua vaativat ainakin

- käyttäjätapahtumien ja järjestelmälokien kehitys
- aktiivisen tunnisteen lisäämisen mahdollisuuden rajaaminen vain määrättyjen työasemien selaimille
- RFID-tunnisteen salauksen määrittäminen
- RFID-tunnisteen yksilöllisyyden varmistaminen.

Asiakkaan mukaan mahdolliseen tuotantokäyttöön jatkokehitettävä sovellus tulisi joka tapauksessa käymään ulkopuolisen tahon suorittaman tietoturva-auditoinnin.

8 Asiakaspalaute ja kehitys

SRT-sovelluksen toiminnallisen prototyypin esittelyn asiakkaan asiantuntijoille toteutui joulukuussa 2017. Pääpiirteittäin sovelluksen lähdekoodissa katsottiin olevan hyvin käyttökelpoisia osia ja merkittävänä asiana nähtiin, että käyttäjän tunnisteen tietojen siirron mekaniikka työasemalta hakemistopalvelimelle on testattu. SRT-sovelluksen katsottiin myös istuvan pääpiirteittäin asiakkaan omaksumaan toteutusmalliin, jota käytetään myös muiden sovellusten osalla.

Saadut huomautukset koskivat Java-luokkien ja JSP-sivujen nimeämiskäytäntöä sekä syöttökenttien historiatietojen käsittelyä, mitkä ovat korjattu liitteissä ilmenevään koodiin.

Loppukäyttöä silmällä pitäen SRT-sovelluksen osalta merkittäviä kehitysalueita olisivat

- sovelluksen ulkoasu
- lokitietojen kerääminen
- admin-osuuden lisääminen parametrien muokkaamista varten.

Ulkoasun osalta käytettäväksi voisi tulla Bootstrap, joka korvaisi opinnäytetyössä tehdyn CSS-tiedoston sekä mahdollisesti myös osan HTML-koodin sisällytetystä JavaScript-koodista. Bootstrap on avoimen lähdekoodin projekti, joka sisältää valmiita määrityksiä CSS-elementeille ja JavaScript-toiminnoille [31]. Bootstrapia voi käyttää ladattuna paikallisena resurssina esimerkiksi sovelluksen web-palvelimella tai vaihtoehtoisesti tämän voi linkittää suoraan Bootstrapin kehitysympäristöstä, jolloin sivuston saatavilla on aina uusin versio [31].

Tuotantokäyttöön suunnatun sovelluksen hyvin merkittävä ominaisuus on lokitietojen kerääminen. SRT-sovelluksessa käyttäjätapahtumien tallennus on hoidettu testimielessä MySQL-tietokantaan, mutta tämä ominaisuus ei ole toimintavarmuuden puolesta valmis. SRT-sovelluksesta puuttuu kokonaan systeemilokien tallennus, joka on myös välttämätön ominaisuus toiminnan diagnostiikan kannalta.

Tällä hetkellä LDAP- ja MySQL-palvelimien yhteysparametrit ja käyttäjätunnukset sekä LDAP-attribuuttien nimet ovat upotettuina Java-koodiin. Jotta sovellusta voitaisiin räätälöidä joustavasti soveltuvaksi muuttuviin toimintaympäristöihin, tulisi näiden ominaisuuksien muokkausta varten olla admin-osio, joka voisi olla esimerkiksi oma web-sovelluksensa, jolloin tietoturvan kannalta tämän käyttöoikeuksien ja toimintaympäristön voisi olla helpommin hallittavissa.

9 Loppusanat

Oppimiskokemuksena SRT-sovelluksen kehittäminen oli itselleni hyvin merkittävä. Vaikkakin lähtökohtainen vaatimusmäärittely oli sinällään suhteellisen yleispiirteinen, niin hahmotelma kehitettävän sovelluksen lopputulemasta oli itselläni alusta saakka kirkkaana mielessä. Itse asiassa koin toteutustavan vapauden helpottavana asiana, koska minulla ei ollut aiempaa kokemusta web-sovellusten ohjelmoinnista, ja näin ollen pystyin soveltamaan etenemistapaa sekä tekniikoita, joihin itselläni oli helpoin mukautua.

Kehitystyön edetessä ja ymmärrykseni lisääntyessä osasin esittää asiakkaan edustajille tarkentavia kysymyksiä sovelluksen toiminnan tavoitteiden yksityiskohdista sekä etenkin hakemistopalvelimen ominaisuuksista. Tämän osalta yhteistyö toimi hyvin joustavasti. Joissain asioissa olisin voinut kysyä asiantuntijoiden mielipidettä jo aiemmin, jolloin olisin varmasti saanut kevennettyä aihepiirien tutkimiseen käyttämääni aikaa.

Asiakkaan näkemyksen mukaan sovelluksen kehitys voisi mahdollisuuksien mukaan jatkua tulevien opinnäytetöiden parissa.

Lähteet

1 TechTarget. 2017. Verkkodokumentti. <http://searchsoftwarequality.techtarget.com/definition/Web-application-Web-app>. Luettu 8.12.2017.

2 Ben Shapiro. 16.4.2013. Verkkodokumentti. <https://seguetech.com/website-vs-web-application-whats-the-difference/>. Luettu 8.12.2017.

3 Nedap. 2018. Verkkodokumentti. <http://www.nedapidentification.com/news/insights/understanding-the-confusing-world-of-rfid-tags-and-readers-in-access-control.html>. Luettu 6.3.2018.

4 HID Global. 2018. Verkkodokumentti. <https:// hidglobal.com/about>. Luettu 28.2.2018.

5 Microsoft. 20.4.2017. Verkkodokumentti. <https://docs.microsoft.com/en-us/windows-hardware/drivers/hid/>. Luettu 28.2.2018.

6 Amanda DeRoo. 3.7.2017. Verkkodokumentti. <https://www.l-trondirect.com/blog/barcode-scanners-a-comparison-of-serial-keyboard-wedge-and-usb-interfaces>. Luettu 28.2.2018.

7 The Apache Software Foundation. 2017. Verkkodokumentti. <https://tomcat.apache.org/whichversion.html>. Luettu 28.11.2017.

8 Eclipse Foundation Inc. 2017. Verkkodokumentti. <https://eclipse.org/org/>. Luettu 8.12.2017.

9 Eclipse Foundation Inc. 2017. Verkkodokumentti. <https://eclipse.org/downloads/eclipse-packages/>. Luettu 8.12.2017.

10 TechTarget. 2017. Verkkodokumentti. <http://theserverside.com/definition/Tomcat>. Luettu 11.12.2017.

11 The Apache Software Foundation. 2017. Verkkodokumentti. <https://tomcat.apache.org/heritage.html>. Luettu 11.12.2017.

12 TechTarget. 2017. Verkkodokumentti. <http://theserverside.com/definition/Java-Runtime-Environment-JRE>. Luettu 13.12.2017.

13 Oracle Corporation. 2017. Verkkodokumentti. <https://docs.oracle.com/cd/E19396-01/817-7619/intro.html>. Luettu 29.11.2017.

14 Perttu Tolvanen. 29.4.2011. Verkkodokumentti. <https://intranet-ostaja-nopas.fi/2011/04/29/kasitteet-ojennukseen-active-directory-ad-ldap-sso-ja-identiteetinhallinta/>. Luettu 29.11.2017.

15 UnboundID. 2015. Verkkodokumentti. < https://www.ldap.com/choosing-an-ldap-server >. Luettu 29.11.2017.

16 OpenLDAP Foundation. 2017. Verkkodokumentti. <https://openldap.org/>. Luettu 29.11.2017.

17 OpenLDAP Foundation. 2017. Verkkodokumentti. <https://openldap.org/foundation/>. Luettu 29.11.2017.

18 UnboundID. 2015. Verkkodokumentti. < https://www.ldap.com/understanding-ldap-schema>. Luettu 13.3.2018.

19 Oracle Corporation. 2018. Verkkodokumentti. < https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html>. Luettu 15.2.2018.

20 Brian Boucheron. 1.6.2017. Verkkodokumentti. <https://digitalocean.com/community/tutorials/how-to-install-and-configure-openldap-and-phpldapadmin-on-ubuntu-16-04>. Luettu 13.3.2018.

21 Oracle Corporation. 2010. Verkkodokumentti. <https://docs.oracle.com/javaee/5/tutorial/doc/bnafe.html>. Luettu 13.12.2017.

22 TheJavaGeek. 2017. Verkkodokumentti. <http://thejava-geek.com/2013/10/27/servlet-listeners-httpsessionlistener-example/>. Luettu 14.12.2017.

23 JournalDev. 2017. Verkkodokumentti. <https://www.journaldev.com/1933/java-servlet-filter-example-tutorial>. Luettu 14.12.2017.

24 TheJavaGeek. 2017. Verkkodokumentti. <http://thejava-geek.com/2013/10/27/servlet-listeners-httpsessionlistener-example/>. Luettu 14.12.2017.

25 JournalDev. 2017. Verkkodokumentti. <https://journaldev.com/1933/java-servlet-filter-example-tutorial>. Luettu 14.12.2017.

26 Oracle Corporation. 2010. Verkkodokumentti. <https://docs.oracle.com/javaee/5/tutorial/doc/bnagx.html>. Luettu 4.1.2018.

27 Refsnes Data. 2018. Verkkodokumentti. <https://w3schools.com/html/html_intro.asp>. Luettu 4.1.2018.

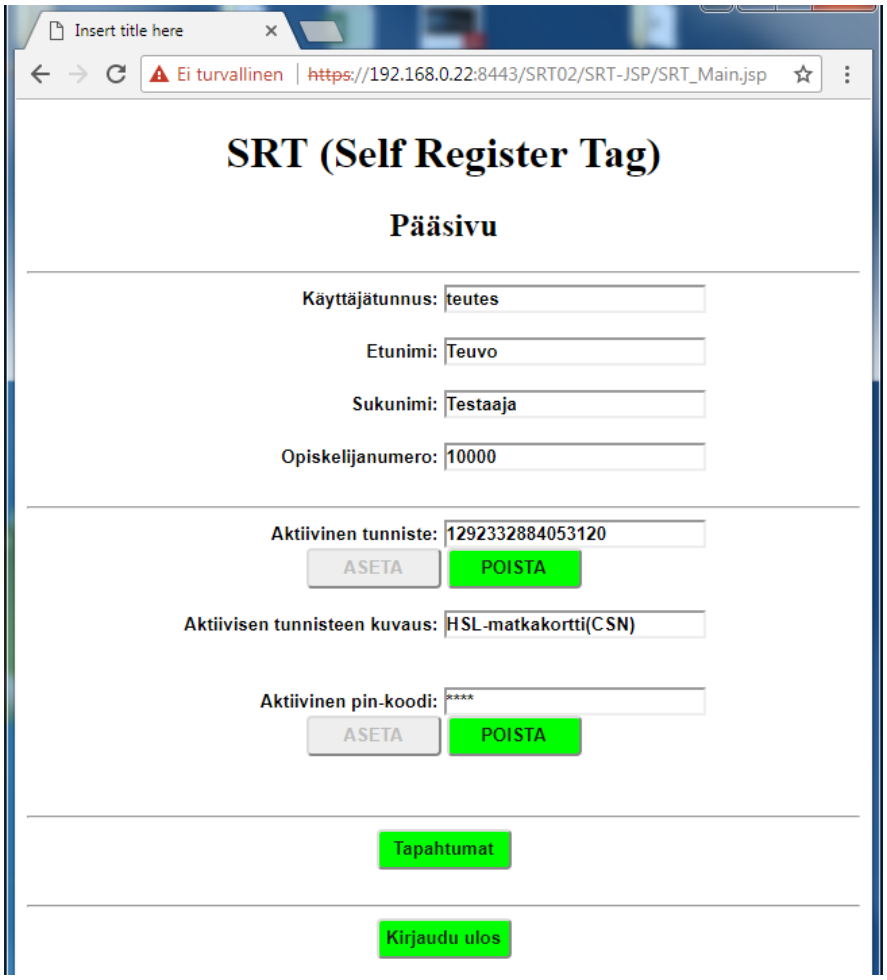
28 Refsnes Data. 2018. Verkkodokumentti. <https://w3schools.com/html/html_css.asp>. Luettu 4.1.2018.

29 Tutorials Point Pvt Ltd. 2018. Verkkodokumentti. <https://tutorialspoint.com/javascript/javascript_overview.htm>. Luettu 2.2.2018.

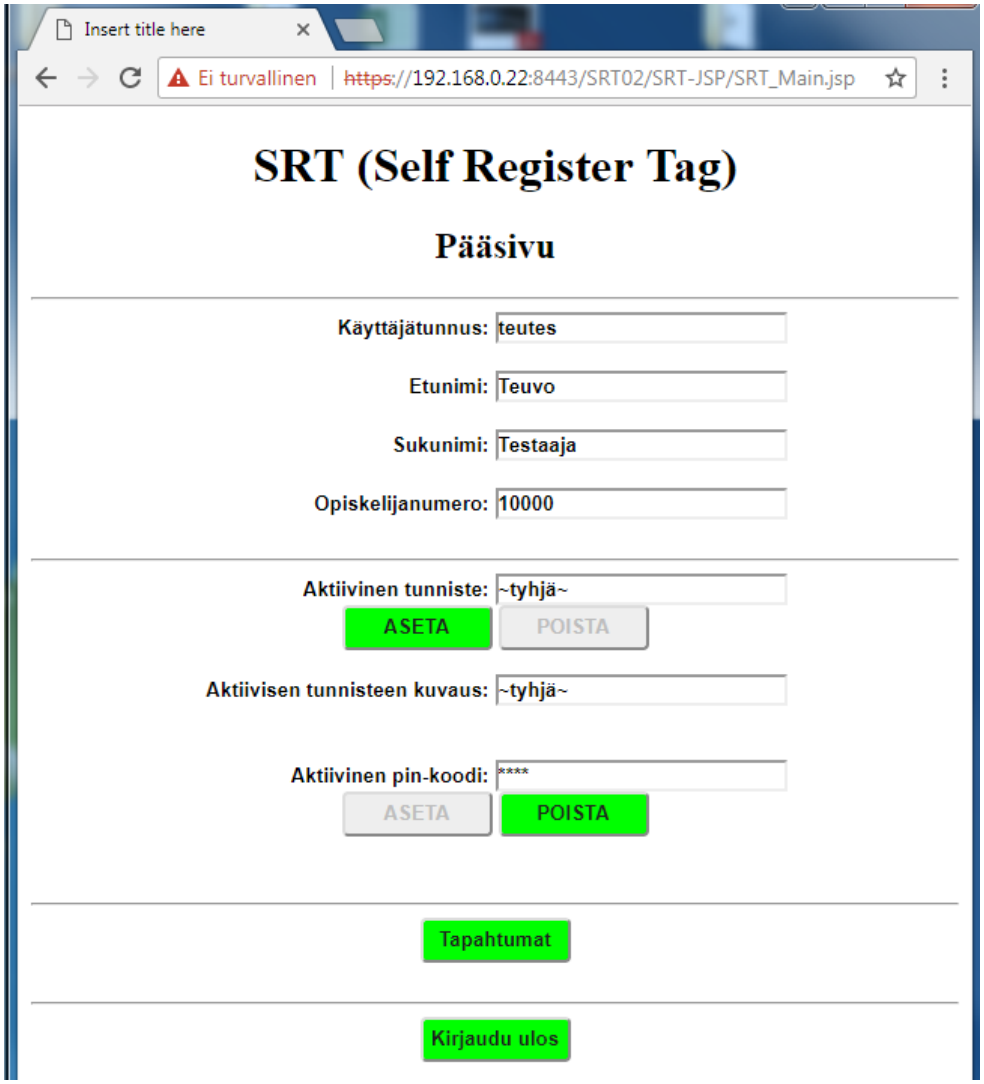
30 Oracle Corporation. 2018. Verkkodokumentti. <https://docs.oracle.com/cd/E13222_01/wls/docs81/webapp/basics.html#146561>. Luettu 15.2.2018.

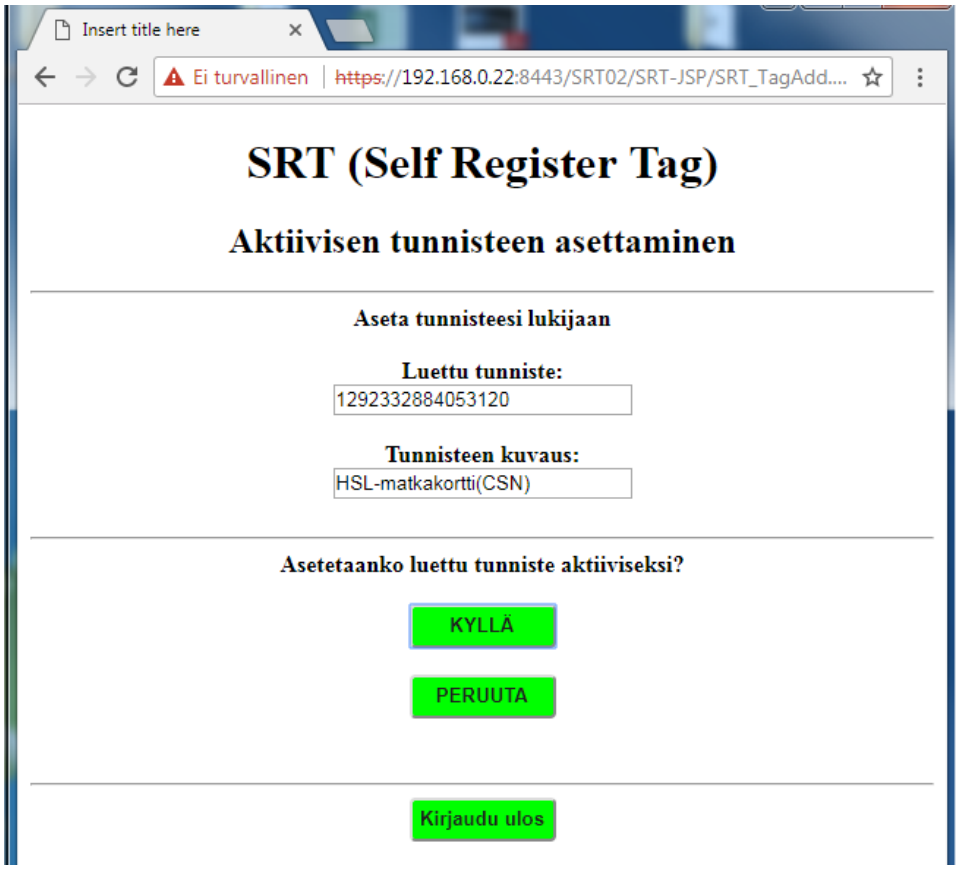
31 Refsnes Data. 2018. Verkkodokumentti. <https://www.w3schools.com/bootstrap/bootstrap_get_started.asp>. Luettu 9.3.2018.

SRT-sovelluksen sivujen selainnäkymät




Insert title here
Ei turvallinen
https://192.168.0.22:8443/SRT02/SRT-JSP/SRT_TagRem...
SRT (Self Register Tag)
Aktiivisen tunnisteen poistaminen
Haluatko varmasti poistaa aktiivisen tunnisteesi?
KYLLÄ
PERUUTA
Kirjaudu ulos


Insert title here
Ei turvallinen
https://192.168.0.22:8443/SRT02/SRT-JSP/SRT_Main.jsp
SRT (Self Register Tag)
Pääsivu
Käyttäjätunnus: teutes
Etunimi: Teuvo
Sukunimi: Testaaja
Opiskelijanumero: 10000
Aktiivinen tunniste: ~tyhjä~
ASETA
POISTA
Aktiivisen tunnisteen kuvaus: ~tyhjä~
Aktiivinen pin-koodi: ****
ASETA
POISTA
Tapahtumat
Kirjaudu ulos

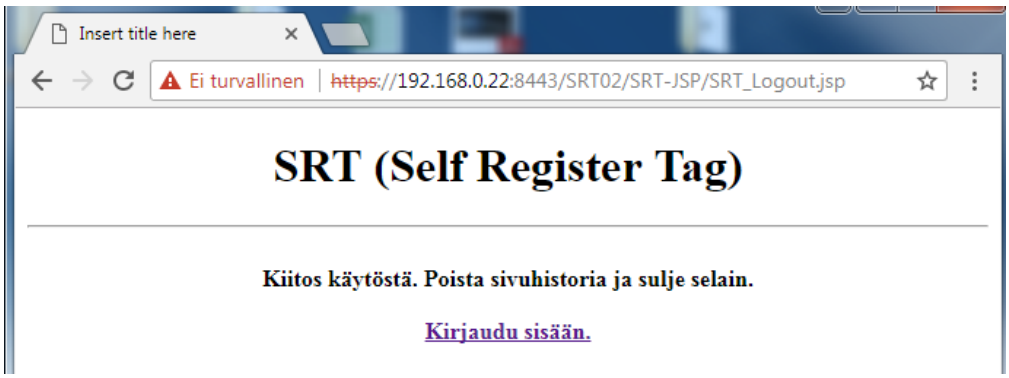

Insert title here
Ei turvallinen
https://192.168.0.22:8443/SRT02/SRT-JSP/SRT_TagAdd....
SRT (Self Register Tag)
Aktiivisen tunnisteen asettaminen
Aseta tunnisteesi lukijaan
Luettu tunniste:
1292332884053120
Tunnisteen kuvaus:
HSL-matkakortti(CSN)
Asetetaanko luettu tunniste aktiiviseksi?
KYLLÄ
PERUUTA
Kirjaudu ulos


Insert title here
Ei turvallinen
https://192.168.0.22:8443/SRT02/SRT-JSP/SRT_PinRem...
SRT (Self Register Tag)
Aktiivisen PIN-koodin poistaminen
Haluatko varmasti poistaa aktiivisen PIN-koodisi
KYLLÄ
PERUUTA
Kirjaudu ulos

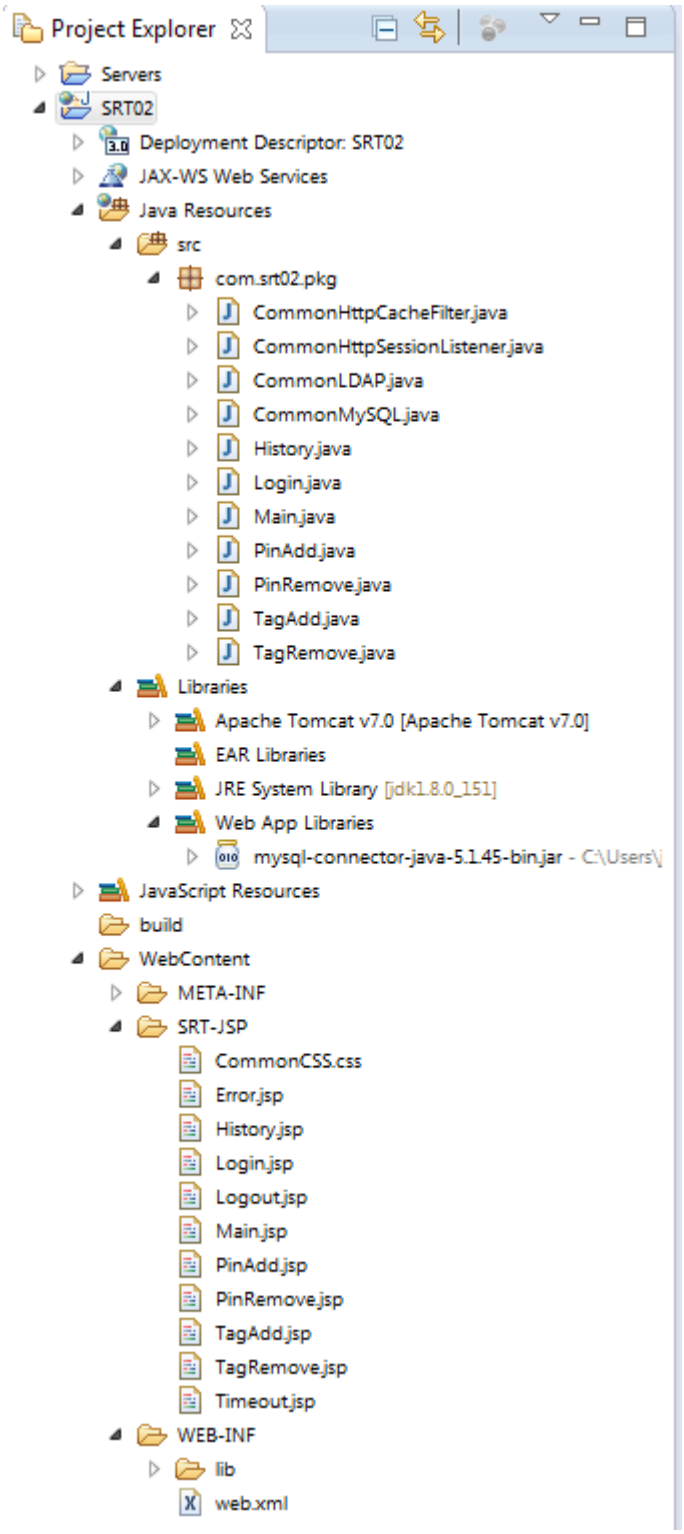

Insert title here
Ei turvallinen
https://192.168.0.22:8443/SRT02/SRT-JSP/SRT_Main.jsp
SRT (Self Register Tag)
Pääsivu
Käyttäjätunnus: teutes
Etunimi: Teuvo
Sukunimi: Testaaja
Opiskelijanumero: 10000
Aktiivinen tunniste: 1292332884053120
ASETA
POISTA
Aktiivisen tunnisteen kuvaus: HSL-matkakortti(CSN)
Aktiivinen pin-koodi: ~tyhjä~
ASETA
POISTA
Tapahtumat
Kirjaudu ulos


Insert title here
Ei turvallinen
https://192.168.0.22:8443/SRT02/SRT-JSP/SRT_PinAdd.jsp
SRT (Self Register Tag)
Aktiivisen PIN-koodin asettaminen
Syötä nelinumeroinen PIN-koodi kaksi kertaa.
....
....
Asetetaanko syötetty PIN-koodi aktiiviseksi?
KYLLÄ
PERUUTA
Kirjaudu ulos


Insert title here
Ei turvallinen
https://192.168.0.22:8443/SRT02/SRT-JSP/SRT_History.jsp
SRT (Self Register Tag)
Tapahtumahistoriasi
Käyttäjätunnus: teutes
Etunimi: Teuvo
Sukunimi: Testaaja
Opiskelijanumero: 10000
20 viimeisintä tapahtumaa, viimeisin ylimpänä:
11.12.2017 21:27:52 | Aktiivinen pin-koodi **** poistettu.
11.12.2017 21:27:10 | Aktiivinen tunniste 1292332884053120 lisätty.
11.12.2017 21:25:57 | Aktiivinen tunniste 1292332884053120 poistettu.
11.12.2017 21:07:30 | Aktiivinen pin-koodi **** lisätty.
11.12.2017 21:07:26 | Aktiivinen pin-koodi **** poistettu.
11.12.2017 21:04:24 | Aktiivinen tunniste 1292332884053120 lisätty.
11.12.2017 21:04:19 | Aktiivinen tunniste 1292332884053120 poistettu.
11.12.2017 21:04:17 | Aktiivinen tunniste 1292332884053120 lisätty.
11.12.2017 21:04:11 | Aktiivinen tunniste 1292332884053120 poistettu.
11.12.2017 21:04:08 | Aktiivinen tunniste 1292332884053120 lisätty.
11.12.2017 21:03:55 | Aktiivinen pin-koodi **** lisätty.
11.12.2017 21:03:51 | Aktiivinen pin-koodi **** poistettu.
11.12.2017 21:03:49 | Aktiivinen pin-koodi **** lisätty.
11.12.2017 21:03:44 | Aktiivinen pin-koodi **** poistettu.
11.12.2017 21:00:19 | Aktiivinen pin-koodi **** lisätty.
11.12.2017 21:00:15 | Aktiivinen pin-koodi **** poistettu.
11.12.2017 20:59:13 | Aktiivinen pin-koodi **** lisätty.
11.12.2017 20:59:08 | Aktiivinen pin-koodi **** poistettu.
11.12.2017 20:09:19 | Aktiivinen pin-koodi **** lisätty.
11.12.2017 20:08:53 | Aktiivinen pin-koodi **** poistettu.
PERUUTA
Kirjaudu ulos


Insert title here
Ei turvallinen
https://192.168.0.22:8443/SRT02/SRT-JSP/SRT_Logout.jsp
SRT (Self Register Tag)
Kiitos käytöstä. Poista sivuhistoria ja sulje selain.
Kirjaudu sisään.

Eclipse-projektipuu



Java-luokka: Login.java

```
package com.srt02.pkg;

import javax.servlet.annotation.WebServlet;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebServlet("/Login")
public class Login extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public Login() {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpS-
ervletResponse response) throws ServletException, IOException {
    }

    protected void doPost(HttpServletRequest request, HttpS-
ervletResponse response) throws ServletException, IOException {

        //Haetaan kirjautumissivulta käyttäjätunnus ja salasana
        String username = request.getParameter("username");
        String password = request.getParameter("password");

        //Asetetaan http-istunnon käyttäjätunnus (= ldap UID) ja sala-
sana (= ldap password)
        //Huom. yleinen istunnon timeout on määritetty web.xml-tiedos-
tossa
        HttpSession session=request.getSession();
        session.setAttribute("username",username);
        session.setAttribute("password",password);

        CommonLDAP common = new CommonLDAP();
        //Ajetaan Common-luokan metodi LdapLoginSearchAttributes,
minkä myötä siirrytään Main-sivulle
        common.LdapLoginSearchAttributes(request, response);
    }
}
```

Java-luokka: Main.java

```
package com.srt02.pkg;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebServlet("/Main")
public class Main extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public Main() {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpS-
ervletResponse response) throws ServletException, IOException {
    }

    protected void doPost(HttpServletRequest request, HttpS-
ervletResponse response) throws ServletException, IOException {

        //Alustetaan Main-sivulla mahdollisesti näkynyt virheviesti
tyhjäksi
        request.getSession().setAttribute("errorMainTagAdd", null);

        HttpSession session = request.getSession(false);
        //Jos istunto on päättynyt, niin siirrytään timeout-sivulle
        if(session == null || session.getAttribute("username") ==
null){
            session=request.getSession();
            response.sendRedirect(request.getContextPath() + "/SRT-
JSP/Timeout.jsp");
            request.getSession().invalidate();
        } else {
            //Muuten jatketaan painetun painikkeen ohjaamalle sivulle
(Logout, TagAdd, TagRemove, PinAdd, PinRemove)
            if (request.getParameter("btnLogout") != null){
                response.sendRedirect(request.getContextPath() +
"/SRT-JSP/Logout.jsp");
                request.getSession().invalidate();
            } else if (request.getParameter("btnTagAdd") != null) {
                response.sendRedirect(request.getContextPath() +
"/SRT-JSP/TagAdd.jsp");
            } else if (request.getParameter("btnTagRemove") != null) {
                response.sendRedirect(request.getContextPath() +
"/SRT-JSP/TagRemove.jsp");
            } else if (request.getParameter("btnPinAdd") != null) {
                response.sendRedirect(request.getContextPath() +
"/SRT-JSP/PinAdd.jsp");
            } else if (request.getParameter("btnPinRemove") != null) {
```

```
                response.sendRedirect(request.getContextPath() +
"/SRT-JSP/PinRemove.jsp");
            } else if (request.getParameter("btnHistory") != null) {
                response.sendRedirect(request.getContextPath() +
"/SRT-JSP/History.jsp");
                CommonMySQL MySQLUserLog = new CommonMySQL();
                //Ajetaan MySQLUserLog-luokan metotodi MySQLSelect
                MySQLUserLog.MySQLSelect(request);
            }
        }
    }
}
```

Java-luokka: PinAdd.java

```
package com.srt02.pkg;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebServlet("/PinAdd")
public class PinAdd extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public PinAdd() {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpS-
ervletResponse response) throws ServletException, IOException {
    }

    protected void doPost(HttpServletRequest request, HttpS-
ervletResponse response) throws ServletException, IOException {

        CommonLDAP common = new CommonLDAP();

        //Jos istunto on päättynyt, niin siirrytään Timeout-sivulle
        HttpSession session=request.getSession(false);
        if(session == null || session.getAttribute("username") ==
null){
            session=request.getSession();
            response.sendRedirect(request.getContextPath() + "/SRT-
JSP/Timeout.jsp");
            request.getSession().invalidate();
        } else {
            //..muuten jatketaan painetun painikkeen ohjaamaan toimin-
toon/sivulle
            try{
                if (request.getParameter("btnLogout") != null){
                    response.sendRedirect(request.getContextPath() +
"/SRT-JSP/Logout.jsp");
                    request.getSession().invalidate();
                } else if (request.getParameter("btnPinAddCancel") !=
null){
                    //Ajetaan Common-luokan metodi LdapSearchAttribu-
tes, jonka myötä siirrytään Main-sivulle
                    common.LdapSearchAttributes(request, response);
                } else if (request.getParameter("btnPinAddYes") !=
null) {
```

```
                    //Ajetaan Common-luokan metodi LdapPinAdd, joka
asettaa käyttäjälle aktiivisen pin-koodin
                    common.LdapPinAdd(request, response);
                    //Ajetaan Common-luokan metodi LdapSearchAttribu-
tes, jonka myötä siirrytään Main-sivulle
                    common.LdapSearchAttributes(request, response);
                }
            } catch (Exception ex){
                    request.getSession().setAttribute("error", "Ei yh-
teyttä palveluun. Yritä myöhemmin uudelleen.");
                    response.sendRedirect(request.getContextPath() +
"/SRT-JSP/Login.jsp");
                    request.getSession().invalidate();
                    ex.printStackTrace();
            }
        }
    }
}
```

Java-luokka: PinRemove.java

```
package com.srt02.pkg;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebServlet("/PinRemove")
public class PinRemove extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public PinRemove() {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpS-
ervletResponse response) throws ServletException, IOException {
    }

    protected void doPost(HttpServletRequest request, HttpS-
ervletResponse response) throws ServletException, IOException {

        CommonLDAP common = new CommonLDAP();

        //Jos istunto on päättynyt, niin siirrytään Timeout-sivulle
        HttpSession session=request.getSession(false);
        if(session == null || session.getAttribute("username") ==
null){
            session=request.getSession();
            response.sendRedirect(request.getContextPath() + "/SRT-
JSP/Timeout.jsp");
            request.getSession().invalidate();
        } else {
            //..muuten jatketaan painetun painikkeen ohjaamaan toimin-
toon/sivulle
            if (request.getParameter("btnLogout") != null){
                response.sendRedirect(request.getContextPath() +
"/SRT-JSP/Logout.jsp");
                request.getSession().invalidate();
            } else if (request.getParameter("btnPinRemoveCancel") !=
null){
                //Ajetaan Common-luokan metodi LdapSearchAttributes,
jonka myötä siirrytään Main-sivulle
                common.LdapSearchAttributes(request, response);
            } else if (request.getParameter("btnPinRemoveYes") !=
null) {
                //Ajetaan Common-luokan metodi LdapPinRemove, joka
poistaa käyttäjän aktiivisen pin-koodin
                common.LdapPinRemove(request, response);
                //Ajetaan Common-luokan metodi LdapSearchAttributes,
jonka myötä siirrytään Main-sivulle
```

```
                common.LdapSearchAttributes(request, response);
            }
        }
    }
}
```

Java-luokka: TagAdd.java

```
package com.srt02.pkg;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebServlet("/TagAdd")
public class TagAdd extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public TagAdd() {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpS-
ervletResponse response) throws ServletException, IOException {
    }

    protected void doPost(HttpServletRequest request, HttpS-
ervletResponse response) throws ServletException, IOException {

        CommonLDAP common = new CommonLDAP();

        //Jos istunto on päättynyt, niin siirrytään Timeout-sivulle
        HttpSession session=request.getSession(false);
        if(session == null || session.getAttribute("username") ==
null){
            session=request.getSession();
            response.sendRedirect(request.getContextPath() + "/SRT-
JSP/Timeout.jsp");
            request.getSession().invalidate();
        } else {
            //..muuten jatketaan painetun painikkeen ohjaamaan toimin-
toon/sivulle
            if (request.getParameter("btnLogout") != null){
                response.sendRedirect(request.getContextPath() +
"/SRT-JSP/Logout.jsp");
                request.getSession().invalidate();
            } else if (request.getParameter("btnTagAddCancel") !=
null){
                //Ajetaan Common-luokan metodi LdapSearchAttributes,
jonka myötä siirrytään Main-sivulle
                common.LdapSearchAttributes(request, response);
            } else if (request.getParameter("btnTagAddYes") != null) {
                //Ajetaan Common-luokan metodi LdapTagAdd, joka aset-
taa käyttäjälle aktiivisen tunnisteen
                common.LdapTagAdd(request, response);
                //Ajetaan Common-luokan metodi LdapSearchAttributes,
jonka myötä siirrytään Main-sivulle
                common.LdapSearchAttributes(request, response);
```

```
                }
            }
        }
    }
```

Java-luokka: TagRemove.java

```
package com.srt02.pkg;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebServlet("/TagRemove")
public class TagRemove extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public TagRemove() {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpS-
ervletResponse response) throws ServletException, IOException {
    }

    protected void doPost(HttpServletRequest request, HttpS-
ervletResponse response) throws ServletException, IOException {

        CommonLDAP common = new CommonLDAP();

        //Jos istunto on päättynyt, niin siirrytään Timeout-sivulle
        HttpSession session=request.getSession(false);
        if(session == null || session.getAttribute("username") ==
null){
            session=request.getSession();
            response.sendRedirect(request.getContextPath() + "/SRT-
JSP/Timeout.jsp");
            request.getSession().invalidate();
        } else {
            //..muuten jatketaan painetun painikkeen ohjaamaan toimin-
toon/sivulle
            if (request.getParameter("btnLogout") != null){
                response.sendRedirect(request.getContextPath() +
"/SRT-JSP/Logout.jsp");
                request.getSession().invalidate();
            } else if (request.getParameter("btnTagRemoveCancel") !=
null){
                //Ajetaan Common-luokan metodi LdapSearchAttributes,
jonka myötä siirrytään Main-sivulle
                common.LdapSearchAttributes(request, response);
            } else if (request.getParameter("btnTagRemoveYes") !=
null){
                //Ajetaan Common-luokan metodi LdapTagRemove, joka
poistaa käyttäjän aktiivisen tunnisteen
                common.LdapTagRemove(request, response);
                //Ajetaan Common-luokan metodi LdapSearchAttributes,
jonka myötä siirrytään Main-sivulle
```

```
                common.LdapSearchAttributes(request, response);
            }
        }
    }
}
```

Java-luokka: History.java

```
package com.srt02.pkg;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebServlet("/History")
public class History extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public History() {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpS-
ervletResponse response) throws ServletException, IOException {
    }

    protected void doPost(HttpServletRequest request, HttpS-
ervletResponse response) throws ServletException, IOException {

        CommonLDAP common = new CommonLDAP();

        //Jos istunto on päättynyt, niin siirrytään Timeout-sivulle
        HttpSession session=request.getSession(false);
        if(session == null || session.getAttribute("username") ==
null){
            session=request.getSession();
            response.sendRedirect(request.getContextPath() + "/SRT-
JSP/Timeout.jsp");
            request.getSession().invalidate();
        } else {
            //..muuten jatketaan painetun painikkeen ohjaamaan toimin-
toon/sivulle
            if (request.getParameter("btnLogout") != null){
                response.sendRedirect(request.getContextPath() +
"/SRT-JSP/Logout.jsp");
                request.getSession().invalidate();
            } else if (request.getParameter("btnHistoryCancel") !=
null){
                //Ajetaan Common-luokan metodi LdapSearchAttributes,
jonka myötä siirrytään Main-sivulle
                common.LdapSearchAttributes(request, response);
            }
        }
    }
}
```

Java-luokka: CommonHttpSessionListener

```
package com.srt02.pkg;

import javax.servlet.annotation.WebListener;
import javax.servlet.http.HttpSessionEvent;
import javax.servlet.http.HttpSessionListener;

@WebListener
public class CommonHttpSessionListener implements HttpSessionListener
{

    public CommonHttpSessionListener() {
    }

    //Listener on tässä muodossa tarkoitettu istuntojen tarkkailuun
testausvaiheessa, eikä sillä ole varsinaista merkitystä sivuston toi-
minnallisuuteen

    //Istunnon aloituksen (created) printtaus konsolin riville
    public void sessionCreated(HttpSessionEvent event) {
        System.out.println("Session created, id: "+event.getSes-
sion().getId());
    }

    //Istunnon aloituksen (created) printtaus konsolin riville
    public void sessionDestroyed(HttpSessionEvent event) {
        System.out.println("Session destroyed, id: "+event.getSes-
sion().getId());
    }
}
```

Java-luokka: CommonHttpCacheFilter

```
package com.srt02.pkg;

import java.io.IOException;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.HttpServletResponse;

@WebFilter("/CommonHttpCacheFilter")
public class CommonHttpCacheFilter implements Filter {

    public CommonHttpCacheFilter() {
    }

    public void destroy() {
    }


    //Kun istunto päättyy (timeout tai uloskirjautuminen) niin selai-
men välimuistiin ei jätetä tietoja, jotka voisivat näkyä esim. Main-
sivulla,
    //mikäli tähän siirrytään selaimen paluu-painikkeella.
    //Filtteri on määritetty web.xml-tiedostossa käytettäväksi kai-
killa sivuilla (/*).
    public void doFilter(ServletRequest request, ServletResponse re-
sponse, FilterChain chain) throws IOException, ServletException {
        HttpServletResponse httpResponse = (HttpServletResponse) re-
sponse;
        httpResponse.setHeader("Cache-Control", "no-cache, no-store,
must-revalidate");
        httpResponse.setHeader("Pragma", "no-cache");
        httpResponse.setDateHeader("Expires", 0);

        chain.doFilter(request, response);
    }

    public void init(FilterConfig fConfig) throws ServletException {
    }
}
```

Java-luokka: CommonLDAP

```
package com.srt02.pkg;

import java.io.IOException;
import java.util.Hashtable;
import java.util.Objects;

import javax.naming.AuthenticationException;
import javax.naming.Context;
import javax.naming.NamingEnumeration;
import javax.naming.NamingException;
import javax.naming.directory.Attributes;
import javax.naming.directory.BasicAttribute;
import javax.naming.directory.ModificationItem;
import javax.naming.directory.SearchControls;
import javax.naming.directory.SearchResult;
import javax.naming.ldap.InitialLdapContext;
import javax.naming.ldap.LdapContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public final class CommonLDAP{

    //Ldap-yhteysparametrien asetus
    private final String base = "ou=People,dc=example,dc=com";
    private final String ldapURL = "ldap://192.168.0.10:389";
    //Ldap-systeemikäyttäjän tunnuksen ja salasanan asetus
    private final String sysUser = "admin";
    private final String sysPassword = "salasana";
    //Tunnistenumeron, tunnisteen kuvauksen ja pin-koodin muuttujiin
asetetaan ldap:n vastaavat attribuutit
    private final String ldapTagNumber = "telephoneNumber";
    private final String ldapTagDescription = "title";
    private final String ldapPinCode = "telexNumber";


    //Metodi, joka ajetaan siirryttäessä Login-sivulta Main-sivulle ja
millä haetaan käyttäjän ldap-attribuutit
    //Virhetilanteissa siirrytään takaisin Login sivulle ja lisätään
error-viesti
    protected void LdapLoginSearchAttributes (HttpServletRequest re-
quest, HttpServletResponse response) throws ServletException, IOExcep-
tion{

        //Alustetaan ldap:sta haettavia attribuutteja vastaavat muut-
tujat
        String firstname = "";
        String lastname = "";
        String idnumber = "";
        String tagnumber = "";
        String tagdescription = "";
        String pincode = "";

        //Haetaan istunnon username ja password
```

```
        HttpSession session=request.getSession();
        String username = (String)session.getAttribute("username");
        String password = (String)session.getAttribute("password");

        //Istunnon käyttäjätunnukseen perustuen asetetaan dn, jolla
suoritetaan haku ldap:sta
        String dn = "uid="+username+","+base;

        //Määritetään ldap-yhteyden parametrit
        Hashtable<String, String> environment = new Hashtable<String,
String>();
        environment.put(Context.INITIAL_CONTEXT_FACTORY,
"com.sun.jndi.ldap.LdapCtxFactory");
        environment.put(Context.PROVIDER_URL, ldapURL);
        environment.put(Context.SECURITY_AUTHENTICATION, "simple");
        environment.put(Context.SECURITY_PRINCIPAL, dn);
        environment.put(Context.SECURITY_CREDENTIALS, password);

        //Haetaan ldap:sta attribuuttien arvot. Mikäli attribuutti
puuttuu, niin vastaava muuttuja saa arvon "null"
        try{
            LdapContext ctx = new InitialLdapContext(environment,
null);
            try {
                SearchControls constraints = new SearchControls();
                String[] attrIDs = {"uidNumber",
                        "givenName",
                        "sn",
                        ldapTagNumber,
                        ldapPinCode,
                        ldapTagDescription};
                constraints.setReturningAttributes(attrIDs);
                constraints.setSearchScope(SearchControls.SUB-
TREE_SCOPE);
                NamingEnumeration<SearchResult> answer =
ctx.search(base, "uid="+username, constraints);
                if (answer.hasMore()){
                    Attributes attr = ((SearchResult) an-
swer.next()).getAttributes();
                    idnumber = Objects.toString(attr.get("uid-
Number"));
                    firstname = Ob-
jects.toString(attr.get("givenName"));
                    lastname = Objects.toString(attr.get("sn"));
                    tagnumber = Objects.toString(attr.get(ldapTag-
Number));
                    //mikäli pin-koodi löytyy, tämä saa arvon "****",
eli pin-koodi-attribuutin todellista arvoa ei tulla näyttämään sivulla
                    if (Objects.toString(attr.get(ldapPinCode)) !=
"null"){
                        pincode = "****";
                    } else {
                        pincode = "";
                    }
                    tagdescription = Ob-
jects.toString(attr.get(ldapTagDescription));
                }
                ctx.close();
            } catch (Exception ex) {
```

```
                ex.printStackTrace();
            }

            //String muodossa "attribute : value" -> muokataan muuttu-
jaan pelkkä "value"-arvo
            idnumber = idnumber.substring(idnumber.in-
dexOf(":")+1).trim();
            firstname = firstname.substring(firstname.in-
dexOf(":")+1).trim();
            lastname = lastname.substring(lastname.in-
dexOf(":")+1).trim();
            tagnumber = tagnumber.substring(tagnumber.in-
dexOf(":")+1).trim();
            pincode = pincode.substring(pincode.in-
dexOf(":")+1).trim();
            tagdescription = tagdescription.substring(tagdescrip-
tion.indexOf(":")+1).trim();

            //Muuttujien arvojen asetus istunnon attribuutteihin,
joita käytetään Main-sivulla
            request.getSession().setAttribute("username", username);
            request.getSession().setAttribute("firstname", firstname);
            request.getSession().setAttribute("lastname", lastname);
            request.getSession().setAttribute("idnumber", idnumber);
            request.getSession().setAttribute("tagnumber", tagnumber);
            request.getSession().setAttribute("tagdescription", tagde-
scription);
            request.getSession().setAttribute("pincode", pincode);

            //Jos kaikki ok - siirtyminen Main-sivulle
            response.sendRedirect(request.getContextPath() + "/SRT-
JSP/Main.jsp");

        //Mikäli kirjautusmitunnukset ovat virheelliset, niin jäädään
Login-sivulle, kentät tyhjennetään ja annetaan virheilmoitus
        } catch (AuthenticationException ex){
            request.getSession().setAttribute("errorLogin", "Virheel-
linen käyttäjätunnus tai salasana!");
            response.sendRedirect(request.getContextPath() + "/SRT-
JSP/Login.jsp");
            ex.printStackTrace();
        //Mikäli ldap-palvelimen yhteyttä ei saada muodostettua, niin
jäädään Login-sivulle, kentät tyhjennetään ja annetaan virheilmoitus
        } catch (NamingException ex){
            request.getSession().setAttribute("errorLogin", "Ei yh-
teyttä autentikointipalveluun. Yritä hetken kuluttua uudelleen.");
            response.sendRedirect(request.getContextPath() + "/SRT-
JSP/Login.jsp");
            ex.printStackTrace();
        }
    }

    //Metodi, joka ajetaan siirryttäessä Main-sivulle ja millä haetaan
käyttäjän ldap-attribuutit
    //Tätä käytetään doPost-toiminnoissa servleteissä(sivuilla): Pi-
nAdd, PinRemove, TagAdd, TagRemove
    //Virhetilanteissa siirrytään Error-sivulle
    protected void LdapSearchAttributes (HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException{
```

```
        //Alustetaan ldap:sta haettavia attribuutteja vastaavat muut-
tujat
        String firstname = "";
        String lastname = "";
        String idnumber = "";
        String tagnumber = "";
        String tagdescription = "";
        String pincode = "";

        //Haetaan istunnon username ja password
        HttpSession session=request.getSession();
        String username = (String)session.getAttribute("username");
        String password = (String)session.getAttribute("password");

        //Istunnon käyttäjätunnukseen perustuen asetetaan dn, jolla
suoritetaan haku ldap:sta
        String dn = "uid="+username+","+base;

        //Määritetään ldap-yhteyden parametrit
        Hashtable<String, String> environment = new Hashtable<String,
String>();
        environment.put(Context.INITIAL_CONTEXT_FACTORY,
"com.sun.jndi.ldap.LdapCtxFactory");
        environment.put(Context.PROVIDER_URL, ldapURL);
        environment.put(Context.SECURITY_AUTHENTICATION, "simple");
        environment.put(Context.SECURITY_PRINCIPAL, dn);
        environment.put(Context.SECURITY_CREDENTIALS, password);

        //Haetaan ldap:sta attribuuttien arvot. Mikäli attribuutti
puuttuu, niin vastaava muuttuja saa arvon "null"
        try{
            LdapContext ctx = new InitialLdapContext(environment,
null);
            try {
                SearchControls constraints = new SearchControls();
                String[] attrIDs = {"uidNumber",
                        "givenName",
                        "sn",
                        ldapTagNumber,
                        ldapPinCode,
                        ldapTagDescription};
                constraints.setReturningAttributes(attrIDs);
                constraints.setSearchScope(SearchControls.SUB-
TREE_SCOPE);
                NamingEnumeration<SearchResult> answer =
ctx.search(base, "uid="+username, constraints);
                if (answer.hasMore()){
                    Attributes attr = ((SearchResult) an-
swer.next()).getAttributes();
                    idnumber = Objects.toString(attr.get("uid-
Number"));
                    firstname = Ob-
jects.toString(attr.get("givenName"));
                    lastname = Objects.toString(attr.get("sn"));
                    tagnumber = Objects.toString(attr.get(ldapTag-
Number));
                    //mikäli pin-koodi löytyy, tämä saa arvon "****",
eli pin-koodi-attribuutin todellista arvoa ei tulla näyttämään sivulla
```

```
                    if (Objects.toString(attr.get(ldapPinCode)) !=
"null"){
                        pincode = "****";
                    } else {
                        pincode = "";
                    }
                    tagdescription = Ob-
jects.toString(attr.get(ldapTagDescription));
                }
                ctx.close();
            } catch (Exception ex) {
                ex.printStackTrace();
            }

            //String muodossa "attribute : value" -> muokataan muuttu-
jaan pelkkä "value"-arvo
            idnumber = idnumber.substring(idnumber.in-
dexOf(":")+1).trim();
            firstname = firstname.substring(firstname.in-
dexOf(":")+1).trim();
            lastname = lastname.substring(lastname.in-
dexOf(":")+1).trim();
            tagnumber = tagnumber.substring(tagnumber.in-
dexOf(":")+1).trim();
            pincode = pincode.substring(pincode.in-
dexOf(":")+1).trim();
            tagdescription = tagdescription.substring(tagdescrip-
tion.indexOf(":")+1).trim();

            //Muuttujien arvojen asetus istunnon attribuutteihin,
joita käytetään Main-sivulla
            request.getSession().setAttribute("username", username);
            request.getSession().setAttribute("firstname", firstname);
            request.getSession().setAttribute("lastname", lastname);
            request.getSession().setAttribute("idnumber", idnumber);
            request.getSession().setAttribute("tagnumber", tagnumber);
            request.getSession().setAttribute("tagdescription", tagde-
scription);
            request.getSession().setAttribute("pincode", pincode);

            /*
            CommonMySQL MySQLUserLog = new CommonMySQL();
            //Ajetaan MySQLUserLog-luokan metotodi MySQLSelect
            MySQLUserLog.MySQLSelect(username);
            */

            //Jos kaikki ok - siirtyminen Main-sivulle
            response.sendRedirect(request.getContextPath() + "/SRT-
JSP/Main.jsp");

        //Mikäli tapahtuu virhe käsittelyn aikana (esim. ldap-serverin
yhteyden timeout), niin tapahtuu ohjaus Error-sivulle
        } catch (NamingException ex){
            response.sendRedirect(request.getContextPath() + "/SRT-
JSP/Error.jsp");
            request.getSession().invalidate();
            ex.printStackTrace();
        }
    }
```

```
    protected void LdapPinAdd (HttpServletRequest request, HttpS-
ervletResponse response) throws ServletException, IOException{

        HttpSession session=request.getSession();

        //Haetaan istunnon käyttäjätunnus, jolle tehdään muutokset
(lisätään pin-koodi)
        String username = (String)session.getAttribute("username");

        //Istunnon käyttäjätunnukseen perustuen asetetaan dn, jolla
määritetään ldap:issa käyttäjä, johon muutokset kohdistuvat
        String dn = "uid="+username+","+base;

        //Haetaan uusi pin-koodi PinAdd-sivulta
        String newPincode = request.getParameter("newPin1");

        //Varmistetaan vielä, että uusi pin-koodi vastaa vaatimuksia
(4 numeroa)
        if(newPincode.length() == 4 && newPincode.matches("^[0-
9]*$")){
            //Jos pin-koodi on ok, jatketaan tämän asettamisella ldap-
attribuuttiin

            //Määritetään ldap-yhteyden parametrit pin-koodin asetta-
mista varten (systeemikäyttäjänä)
            Hashtable<String, String> modify = new Hashtable<String,
String>();
            modify.put(Context.INITIAL_CONTEXT_FACTORY,
"com.sun.jndi.ldap.LdapCtxFactory");
            modify.put(Context.PROVIDER_URL, ldapURL);
            modify.put(Context.SECURITY_AUTHENTICATION, "simple");
            modify.put(Context.SECURITY_PRINCIPAL, "cn="+sys-
User+",dc=example,dc=com");
            modify.put(Context.SECURITY_CREDENTIALS, sysPassword);

            //Luodaan uusi attribuutti annetulla arvolla
            try{
                LdapContext ctx = new InitialLdapContext(modify,
null);
                ModificationItem[] mods = new ModificationItem[1];
                mods[0] = new ModificationItem(LdapContext.REPLACE_AT-
TRIBUTE, new BasicAttribute(ldapPinCode,newPincode));
                ctx.modifyAttributes(dn, mods);
                ctx.close();

                //Alustetaan PinAdd-toiminnon SQL-kantaan tallennet-
tava tapahtumaviesti
                String sqlLogEvent = "Aktiivinen pin-koodi **** li-
sätty.";
                CommonMySQL MySQLUserLog = new CommonMySQL();
                //Ajetaan MySQLUserLog-luokan metotodi MySQLInsert,
johon asetetaan username ja sqlLogEvent
                MySQLUserLog.MySQLInsert(username, sqlLogEvent);

            } catch (NamingException ex){
                ex.printStackTrace();
            }
        }
```

```
    }

    protected void LdapPinRemove (HttpServletRequest request, HttpS-
ervletResponse response) throws ServletException, IOException{

        HttpSession session=request.getSession();

        //Haetaan istunnon käyttäjätunnus, jolle tehdään muutokset
(poistetaan pin-koodi)
        String username = (String)session.getAttribute("username");

        //Istunnon käyttäjätunnukseen perustuen asetetaan dn, jolla
määritetään ldap:issa käyttäjä, johon muutokset kohdistuvat
        String dn = "uid="+username+","+base;

        //Määritetään ldap-yhteyden parametrit pin-koodin poistamista
varten (systeemikäyttäjänä)
        Hashtable<String, String> modify = new Hashtable<String,
String>();
        modify.put(Context.INITIAL_CONTEXT_FACTORY,
"com.sun.jndi.ldap.LdapCtxFactory");
        modify.put(Context.PROVIDER_URL, ldapURL);
        modify.put(Context.SECURITY_AUTHENTICATION, "simple");
        modify.put(Context.SECURITY_PRINCIPAL, "cn="+sysUser+",dc=ex-
ample,dc=com");
        modify.put(Context.SECURITY_CREDENTIALS, sysPassword);

        //Poistetaan käyttäjän pin-koodia vastaava attribuutti
ldap:sta
        try{
            LdapContext ctx = new InitialLdapContext(modify, null);
            ModificationItem[] mods = new ModificationItem[1];
            mods[0] = new ModificationItem(LdapContext.REPLACE_ATTRIB-
UTE, new BasicAttribute(ldapPinCode));
            ctx.modifyAttributes(dn, mods);
            ctx.close();

            //Alustetaan PinRemove-toiminnon SQL-kantaan tallennettava
tapahtumaviesti
            String sqlLogEvent = "Aktiivinen pin-koodi **** pois-
tettu.";
            CommonMySQL MySQLUserLog = new CommonMySQL();
            //Ajetaan MySQLUserLog-luokan metotodi MySQLInsert, johon
asetetaan username ja sqlLogEvent
            MySQLUserLog.MySQLInsert(username, sqlLogEvent);

        } catch (NamingException ex){
            ex.printStackTrace();
        }
    }

    protected void LdapTagAdd (HttpServletRequest request, HttpS-
ervletResponse response) throws ServletException, IOException{

        HttpSession session=request.getSession();

        //Haetaan istunnon käyttäjätunnus, jolle tehdään muutokset
(lisätään tunnistenumero ja tunnisteen kuvaus)
        String username = (String)session.getAttribute("username");
```

```
        //Istunnon käyttäjätunnukseen perustuen asetetaan dn, jolla
määritetään ldap:issa käyttäjä, johon muutokset kohdistuvat
        String dn = "uid="+username+","+base;

        //Haetaan uusi tunnistenumero ja tunnisteen kuvaus TagAdd-si-
vulta
        String newTagNumber = request.getParameter("newTagNumber");
        String newTagDescription = request.getParameter("newTagDe-
scription");


        //Varmistetaan vielä, että uusi tunnistenumero on pelkkinä nu-
meroina ja max 12 merkkiä sekä tunnisteen kuvaus on max 30 merkkiä
        if(newTagNumber.length() <= 20 && newTagNumber.matches("^[0-
9]*$") && newTagDescription.length() <= 30){
            //Jos arvot ovat ok, niin jatketaan näiden asettamisella
ldap-attribuutteihin

            //Määritetään ldap-yhteyden parametrit arvojen asettamista
varten (systeemikäyttäjänä)
            Hashtable<String, String> modify = new Hashtable<String,
String>();
            modify.put(Context.INITIAL_CONTEXT_FACTORY,
"com.sun.jndi.ldap.LdapCtxFactory");
            modify.put(Context.PROVIDER_URL, ldapURL);
            modify.put(Context.SECURITY_AUTHENTICATION, "simple");
            modify.put(Context.SECURITY_PRINCIPAL, "cn="+sys-
User+",dc=example,dc=com");
            modify.put(Context.SECURITY_CREDENTIALS, sysPassword);

            //Haetaan ldap:sta systeemikäyttäjänä vastaavia tunniste-
numeroita sekä tarkastetaan, että onko käyttäjällä
            //aktiivista tunnistetta
            try {
                LdapContext ctx = new InitialLdapContext(modify,
null);
                String tagnumberAll = "";
                String tagnumberUser = "";
                String tagdescriptionUser = "";

                SearchControls constraints = new SearchControls();
                String[] attrIDs = {ldapTagNumber};
                constraints.setReturningAttributes(attrIDs);
                constraints.setSearchScope(SearchControls.SUB-
TREE_SCOPE);
                NamingEnumeration<SearchResult> answer1 =
ctx.search(base, ldapTagNumber+"="+newTagNumber, constraints);
                NamingEnumeration<SearchResult> answer2 =
ctx.search(base, "uid="+username, constraints);
                if(answer1.hasMore()){
                    Attributes attr = ((SearchResult) an-
swer1.next()).getAttributes();
                    //tallennetaan löytyneen attribuutin arvo tag-
numberAll-muuttujaan. Jos attribuuttia ei löytynyt, tämä saa arvon
"null"
                    tagnumberAll = Objects.toString(attr.get(ldapTag-
Number));
                }
```

```
                if(answer2.hasMore()){
                    Attributes attr = ((SearchResult) an-
swer2.next()).getAttributes();
                    //tallennetaan löytyneen attribuutin arvo tag-
numberUser-muuttujaan. Jos attribuuttia ei löytynyt, tämä saa arvon
"null"
                    tagnumberUser = Objects.toString(attr.get(ldapTag-
Number));
                    tagdescriptionUser = Ob-
jects.toString(attr.get(ldapTagDescription));
                }
                //String muodossa "attribute : value" -> muokataan
muuttujiin pelkkä "value"-arvo
                tagnumberAll = tagnumberAll.substring(tagnumberAll.in-
dexOf(":")+1).trim();
                tagnumberUser = tagnumberUser.substring(tag-
numberUser.indexOf(":")+1).trim();
                tagdescriptionUser = tagdescriptionUser.sub-
string(tagdescriptionUser.indexOf(":")+1).trim();

                //Jos lisättävä tunnistenumero löytyy ldap:sta, pala-
taan Main-sivulle ja esitetään virheviesti
                if(tagnumberAll.equals(newTagNumber)){
                    request.getSession().setAttribute("errorMainTa-
gAdd", "Tunnistenumero on varattu! Yritä toista tunnistetta tai ota
yhteys ylläpitäjään.");
                //Jos käyttäjällä on jokin tunnistenumero ldap:ssa,
palataan Main-sivulle ja esitetään virheviesti
                } else if ((tagnumberUser != "" && tagnumberUser !=
"null") || (tagdescriptionUser != "" && tagdescriptionUser !=
"null")){
                    request.getSession().setAttribute("errorMainTa-
gAdd", "Vain 1 tunniste/käyttäjä! Poista aiempi tunniste lisätäksesi
uuden.");
                //Jos tunnistenumero ei ole käytössä ja käyttäjällä ei
ole aktiivista tunnistetta:
                } else {
                    //Luodaan uudet attribuutit annetuilla arvoilla
(newTagNumber ja newTagDescription)
                    try{
                        ModificationItem[] mods = new Modifica-
tionItem[2];
                        mods[0] = new ModificationItem(LdapContext.RE-
PLACE_ATTRIBUTE, new BasicAttribute(ldapTagNumber,newTagNumber));
//tunnistenumero
                        mods[1] = new ModificationItem(LdapContext.RE-
PLACE_ATTRIBUTE, new BasicAttribute(ldapTagDescription,newTagDescrip-
tion)); //tunnisteen kuvaus
                        ctx.modifyAttributes(dn, mods);
                        ctx.close();

                        //Alustetaan TagAdd-toiminnon SQL-kantaan tal-
lennettava tapahtumaviesti
                        String sqlLogEvent = "Aktiivinen tunniste
"+newTagNumber+" lisätty.";
                        CommonMySQL MySQLUserLog = new CommonMySQL();
                        //Ajetaan MySQLUserLog-luokan metotodi
MySQLInsert, johon asetetaan username ja sqlLogEvent
```

```
                        MySQLUserLog.MySQLInsert(username, sql-
LogEvent);

                    } catch (NamingException ex){
                        ex.printStackTrace();
                    }
                } ctx.close();
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }
    }

    protected void LdapTagRemove (HttpServletRequest request, HttpS-
ervletResponse response) throws ServletException, IOException{

        HttpSession session=request.getSession();

        //Haetaan istunnon käyttäjätunnus, jolle tehdään muutokset
(poistetaan tunnistenumero ja tunnisteen kuvaus)
        String username = (String)session.getAttribute("username");

        //Istunnon käyttäjätunnukseen perustuen asetetaan dn, jolla
määritetään ldap:issa käyttäjä, johon muutokset kohdistuvat
        String dn = "uid="+username+","+base;

        //Määritetään ldap-yhteyden parametrit tunnistenumeron ja tun-
nisteen kuvauksen poistamista varten (systeemikäyttäjänä)
        Hashtable<String, String> modify = new Hashtable<String,
String>();
        modify.put(Context.INITIAL_CONTEXT_FACTORY,
"com.sun.jndi.ldap.LdapCtxFactory");
        modify.put(Context.PROVIDER_URL, ldapURL);
        modify.put(Context.SECURITY_AUTHENTICATION, "simple");
        modify.put(Context.SECURITY_PRINCIPAL, "cn="+sysUser+",dc=ex-
ample,dc=com");
        modify.put(Context.SECURITY_CREDENTIALS, sysPassword);

        try{
            LdapContext ctx = new InitialLdapContext(modify, null);
            String tagnumberUser = "";

            //Haetaan käyttäjän nykyinen tunnistenumero lokitietoa
varte
            SearchControls constraints = new SearchControls();
            String[] attrIDs = {ldapTagNumber};
            constraints.setReturningAttributes(attrIDs);
            constraints.setSearchScope(SearchControls.SUBTREE_SCOPE);
            NamingEnumeration<SearchResult> answer = ctx.search(base,
"uid="+username, constraints);
            if(answer.hasMore()){
                Attributes attr = ((SearchResult) an-
swer.next()).getAttributes();
                //tallennetaan löytyneen attribuutin arvo tagnumbe-
rUser-muuttujaan. Jos attribuuttia ei löytynyt, tämä saa arvon "null"
                tagnumberUser = Objects.toString(attr.get(ldapTag-
Number));
            }
```

```
            //String muodossa "attribute : value" -> muokataan muuttu-
jiin pelkkä "value"-arvo
            tagnumberUser = tagnumberUser.substring(tagnumberUser.in-
dexOf(":")+1).trim();

            //Poistetaan käyttäjän tunnistenumeroa ja tunnisteen ku-
vausta vastaavat attribuutit ldap:sta
            ModificationItem[] mods = new ModificationItem[2];
            mods[0] = new ModificationItem(LdapContext.REPLACE_ATTRIB-
UTE, new BasicAttribute(ldapTagNumber)); //tunnistenumero
            mods[1] = new ModificationItem(LdapContext.REPLACE_ATTRIB-
UTE, new BasicAttribute(ldapTagDescription)); //tunnisteen kuvaus
            ctx.modifyAttributes(dn, mods);
            ctx.close();

            //Alustetaan TagRemove-toiminnon SQL-kantaan tallennettava
tapahtumaviesti
            String sqlLogEvent = "Aktiivinen tunniste "+tagnumbe-
rUser+" poistettu.";
            CommonMySQL MySQLUserLog = new CommonMySQL();
            //Ajetaan MySQLUserLog-luokan metotodi MySQLInsert, johon
asetetaan username ja sqlLogEvent
            MySQLUserLog.MySQLInsert(username, sqlLogEvent);

        } catch (NamingException ex){
            ex.printStackTrace();
        }
    }
}
```

Java-luokka: CommonMySQL.java

```
package com.srt02.pkg;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

public final class CommonMySQL {

    //Alustetaan MySQL-palvelimen yhteysasetukset ja tietokannan kir-
jautumistunnukset
    private final String sqlURL =
"jdbc:mysql://192.168.0.10:3306/srtdb";
    private final String sqlUser = "admin";
    private final String sqlPassword = "salasana";

    protected void MySQLInsert(String username, String event){
        Connection con = null;
        PreparedStatement pst = null;

        try{
            Class.forName("com.mysql.jdbc.Driver");
        } catch (ClassNotFoundException ex) {
            ex.printStackTrace();
        }

        try{
            con = DriverManager.getConnection(sqlURL, sqlUser,
sqlPassword);

            pst = con.prepareStatement("INSERT INTO SrtUser-
Log(Username,Event) VALUES(?,?)");
            pst.setString(1, username);
            pst.setString(2, event);
            pst.executeUpdate();
        } catch (SQLException ex) {
            ex.printStackTrace();
        } finally {
            try{
                if(pst != null){
                    pst.close();
                }
                if(con != null){
                    con.close();
                }
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    }

    protected void MySQLSelect(HttpServletRequest request){
```

```
        //Haetaan istunnon käyttäjätunnus, jota käytetään SQL-haussa
        HttpSession session=request.getSession();
        String username = (String)session.getAttribute("username");

        Connection con = null;
        PreparedStatement pst = null;
        ResultSet rs = null;

        try{
            Class.forName("com.mysql.jdbc.Driver");
        } catch (ClassNotFoundException ex) {
            ex.printStackTrace();
        }

        try{
            con = DriverManager.getConnection(sqlURL, sqlUser,
sqlPassword);

            pst = con.prepareStatement("SELECT DATE_FOR-
MAT(Timestamp,'%d.%m.%Y %H:%i:%s'),Event FROM SrtUserLog WHERE
Username=? ORDER BY Timestamp DESC LIMIT 20");
            pst.setString(1, username);

            rs = pst.executeQuery();

            StringBuilder events = new StringBuilder();
            while (rs.next()){
                events.append(rs.getString("DATE_FOR-
MAT(Timestamp,'%d.%m.%Y %H:%i:%s')")+" | "+rs.getString("Event"));
                events.append("\n");
            }
            request.getSession().setAttribute("events", events);
        } catch (SQLException ex) {
            request.getSession().setAttribute("events", "Tapahtumahis-
toria ei ole tällä hetkellä saatavilla");
            ex.printStackTrace();
        } finally {
            try{
                if(pst != null){
                    pst.close();
                }
                if(con != null){
                    con.close();
                }
                if(rs != null){
                    rs.close();
                }
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    }
}
```

JSP-sivu: Login.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<link rel="stylesheet" href="CommonCSS.css" type="text/css"/>
</head>
<body>
<div><h1>SRT (Self Register Tag)</h1></div>
<div><h2>Sisäänkirjautuminen</h2></div>
<hr>
<form action="Login" method="post">
<div><b>Käyttäjätunnus:</b></div>
<div><input type="text" name="username" id="username"
onkeyup="notEmpty()" style="text-align:left; width: 130px" autocom-
plete="off"></div><br>
<div><b>Salasana:</b></div>
<div><input type="password" name="password" id="password"
onkeyup="notEmpty()" style="text-align:left; width: 130px" autocom-
plete="off"></div><br>
<div><button type="submit" name="btnLogin" id="btnLogin" class="btn-
disabled" disabled>Kirjaudu...</button></div><br>
<div><p style=color:red>${errorLogin}</p></div>
</form>
<script type="text/javascript">
    window.onload = function(){
        var i = document.getElementById("username");
        i.focus();
        i.select();
    }

    function notEmpty(){
        var i1 = document.getElementById("username");
        var i2 = document.getElementById("password");
        if(i1.value == ""|| i2.value == ""){
            document.getElementById("btnLogin").className = "btn-disa-
bled";
            document.getElementById("btnLogin").disabled = true;
        } else {
            document.getElementById("btnLogin").className = "btn-ena-
bled";
            document.getElementById("btnLogin").disabled = false;
        }
    }
</script>
</body>
</html>
```

JSP-sivu: Main.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<link rel="stylesheet" href="CommonCSS.css" type="text/css"/>
</head>
<body>
<div><h1>SRT (Self Register Tag)</h1></div>
<div><h2>Pääsivu</h2></div>
<hr>
<div><input type="text" class="input-text1" value="Käyttäjätunnus:"
readonly>
<input type="text" name="username" value="${username}" class="input-
text2" readonly></div><br>
<div><input type="text" class="input-text1" value="Etunimi:" readonly>
<input type="text" name="firstname" value="${firstname}" class="input-
text2" readonly></div><br>
<div><input type="text" class="input-text1" value="Sukunimi:"
readonly>
<input type="text" name="lastname" value="${lastname}" class="input-
text2" readonly></div><br>
<div><input type="text" class="input-text1" value="Opiskelijanumero:"
readonly>
<input type="text" name="idnumber" value="${idnumber}" class="input-
text2" readonly></div><br>
<hr>
<form action="Main" method="post">
<div><input type="text" class="input-text1" value="Aktiivinen tun-
niste:" readonly>
<input type="text" name="tagnumber" id="tagnumber" value="" class="in-
put-text2" readonly></div>
<div><button type="submit" name="btnTagAdd" id="btnTagAdd" class="btn-
disabled" disabled>ASETA</button>
<button type="submit" name="btnTagRemove" id="btnTagRemove"
class="btn-disabled" disabled>POISTA</button></div>

<div><p style=color:red>${errorMainTagAdd}</p></div>

<div><input type="text" class="input-text1" value="Aktiivisen tunnis-
teen kuvaus:" readonly>
<input type="text" name="tagdescription" id="tagdescription"
class="input-text2" readonly></div><br><br>
<div><input type="text" class="input-text1" value="Aktiivinen pin-
koodi:" readonly>
<input type="text" name="pincode" id="pincode" class="input-text2"
readonly></div>
<div><button type="submit" name="btnPinAdd" id="btnPinAdd" class="btn-
disabled" disabled>ASETA</button>
<button type="submit" name="btnPinRemove" id="btnPinRemove"
class="btn-disabled" disabled>POISTA</button></div><br><br>
<hr>
```

```
<div><button type="submit" name="btnHistory" id="btnHistory"
class="btn-enabled">Tapahtumat</button></div><br>
<hr>
<div><button type="submit" name="btnLogout" id="btnLogout" class="btn-
enabled">Kirjaudu ulos</button></div><br>
</form>
<script type="text/javascript">
    window.onload = function(){

        var i1 = "${tagnumber}";
        var i2 = "${tagdescription}";
        var i3 = "${pincode}";
        if(i1 == "null"){
            document.getElementById("btnTagAdd").className = "btn-ena-
bled";
            document.getElementById("btnTagAdd").disabled = false;
            document.getElementById("btnTagRemove").className = "btn-
disabled";
            document.getElementById("btnTagRemove").disabled = true;
            document.getElementById("tagnumber").value = "~tyhjä~";
            document.getElementById("tagdescription").value =
"~tyhjä~";
        } else {
            document.getElementById("btnTagAdd").className = "btn-dis-
abled";
            document.getElementById("btnTagAdd").disabled = true;
            document.getElementById("btnTagRemove").className = "btn-
enabled";
            document.getElementById("btnTagRemove").disabled = false;
            document.getElementById("tagnumber").value = i1;
            document.getElementById("tagdescription").value = i2;
        }
        if(i3 != "****"){
            document.getElementById("btnPinAdd").className = "btn-ena-
bled";
            document.getElementById("btnPinAdd").disabled = false;
            document.getElementById("btnPinRemove").className = "btn-
disabled";
            document.getElementById("btnPinRemove").disabled = true;
            document.getElementById("pincode").value = "~tyhjä~";
        } else {
            document.getElementById("btnPinAdd").className = "btn-dis-
abled";
            document.getElementById("btnPinAdd").disabled = true;
            document.getElementById("btnPinRemove").className = "btn-
enabled";
            document.getElementById("btnPinRemove").disabled = false;
            document.getElementById("pincode").value = i3;
        }
    };
</script>
</body>
</html>
```

JSP-sivu: PinAdd.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<link rel="stylesheet" href="CommonCSS.css" type="text/css"/>
</head>
<body>
<div><h1>SRT (Self Register Tag)</h1></div>
<div><h2>Aktiivisen PIN-koodin asettaminen</h2></div>
<hr>
<div><b>Syötä nelinumeroinen PIN-koodi kaksi kertaa.</b></div><br>
<form action="PinAdd" method="post">
<div><input type="password" name="newPin1" id="newPin1" pattern="[0-
9]{4}" maxlength="4" title="Neljä numeroa!" onkeyup="notEmpty()"
style="text-align:center; width: 80px" autocomplete="off"></div><br>
<div><input type="password" name="newPin2" id="newPin2" pattern="[0-
9]{4}" maxlength="4" title="Neljä numeroa!" onkeyup="notEmpty()"
style="text-align:center; width: 80px" autocomplete="off"></div><br>
<hr>
<div><b>Asetetaanko syötetty PIN-koodi aktiiviseksi?</b></div><br>
<div><button type="submit" name="btnPinAddYes" id="btnPinAddYes"
class="btn-disabled" disabled>KYLLÄ</button></div><br>
<div><button type="submit" name="btnPinAddCancel" id="btnPinAddCancel"
class="btn-enabled" formnovalidate>PERUUTA</button></div><br><br>
<hr>
<div><button type="submit" name="btnLogout" id="btnLogout" class="btn-
enabled" formnovalidate>Kirjaudu ulos</button></div><br>
</form>
<script type="text/javascript">
    function notEmpty(){
        var i1 = document.getElementById("newPin1");
        var i2 = document.getElementById("newPin2");
        if(i1.value != i2.value){
            document.getElementById("btnPinAddYes").className = "btn-
disabled";
            document.getElementById("btnPinAddYes").disabled = true;
        } else {
            document.getElementById("btnPinAddYes").className = "btn-
enabled";
            document.getElementById("btnPinAddYes").disabled = false;
        }
    }
</script>
</body>
</html>
```

JSP-sivu: PinRemove.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<link rel="stylesheet" href="CommonCSS.css" type="text/css"/>
</head>
<body>
<div><h1>SRT (Self Register Tag)</h1></div>
<div><h2>Aktiivisen PIN-koodin poistaminen</h2></div>
<hr>
<div><b>Haluatko varmasti poistaa aktiivisen PIN-koodisi</b></div><br>
<form action="PinRemove" method="post">
<div><button type=submit name="btnPinRemoveYes" id="btnPinRemoveYes"
class="btn-enabled">KYLLÄ</button></div><br>
<div><button type=submit name="btnPinRemoveCancel" id="btnPinRe-
moveCancel" class="btn-enabled">PERUUTA</button></div><br><br>
<hr>
<div><button type="submit" name="btnLogout" id="btnLogout" class="btn-
enabled" formnovalidate>Kirjaudu ulos</button></div><br>
</form>
</body>
</html>
```

JSP-sivu: TagAdd.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<link rel="stylesheet" href="CommonCSS.css" type="text/css"/>
</head>
<body>
<div><h1>SRT (Self Register Tag)</h1></div>
<div><h2>Aktiivisen tunnisteen asettaminen</h2></div>
<hr>
<div><b>Aseta tunnisteesi lukijaan</b></div><br>
<form action="TagAdd" method="post">
<div><b>Luettu tunniste:</b></div>
<div><input type=text name="newTagNumber" id="newTagNumber" pat-
tern="\d*" maxlength="20" title="Pelkkiä numeroita!"
onkeyup="notEmpty()" oninput="tagDescription()" onclick="tagClear()"
style="width: 200px" autocomplete="off"></div><br>
<div><b>Tunnisteen kuvaus:</b></div>
<div><input type=text name="newTagDescription" id="newTagDescription"
maxlength="30" onkeyup="notEmpty()" style="width: 200px" autocom-
plete="off" readonly></div><br>
<hr>
<div><b>Asetetaanko luettu tunniste aktiiviseksi?</b></div><br>
<div><button type="submit" name="btnTagAddYes" id="btnTagAddYes"
class="btn-disabled" disabled>KYLLÄ</button></div><br>
<div><button type="submit" name="btnTagAddCancel" id="btnTagAddCancel"
class="btn-enabled" formnovalidate>PERUUTA</button></div><br><br>
<hr>
<div><button type="submit" name="btnLogout" id="btnLogout" class="btn-
enabled" formnovalidate>Kirjaudu ulos</button></div><br>
</form>
<script type="text/javascript">
    window.onload = function(){
        var i = document.getElementById("newTagNumber");
        i.focus();
        i.select();
    };

    function tagClear(){
        document.getElementById("newTagNumber").value = "";
        document.getElementById("newTagDescription").value = "";
        document.getElementById("btnTagAddYes").className = "btn-disa-
bled";
    }

    function tagDescription(){
        var i1 = document.getElementById("newTagNumber");
        var i2 = document.getElementById("newTagDescription");
        if(i1.value.substring(0,3) == 129){
            i2.value = "HSL-matkakortti(CSN)";
        } else if(i1.value.substring(0,3) == 125){
            i2.value = "Valttikortti(CSN)";
```

```
        } else {
            i2.value = "Tuntematon korttityyppi(CSN)";
        }
    }

    function notEmpty(){
        var i1 = document.getElementById("newTagNumber");
        var i2 = document.getElementById("newTagDescription");
        if(i1.value == "" || i2.value == ""){
            document.getElementById("btnTagAddYes").className = "btn-
disabled";
            document.getElementById("btnTagAddYes").disabled = true;
        } else {
            document.getElementById("btnTagAddYes").className = "btn-
enabled";
            document.getElementById("btnTagAddYes").disabled = false;
        }
    }
</script>
</body>
</html>
```

JSP-sivu: TagRemove.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<link rel="stylesheet" href="CommonCSS.css" type="text/css"/>
</head>
<body>
<div><h1>SRT (Self Register Tag)</h1></div>
<div><h2>Aktiivisen tunnisteen poistaminen</h2></div>
<hr>
<div><b>Haluatko varmasti poistaa aktiivisen tunnis-
teesi?</b></div><br>
<form action="TagRemove" method="post">
<div><button type=submit name="btnTagRemoveYes" id="btnTagRemoveYes"
class="btn-enabled">KYLLÄ</button></div><br>
<div><button type=submit name="btnTagRemoveCancel" id="btnTagRe-
moveCancel" class="btn-enabled">PERUUTA</button></div><br><br>
<hr>
<div><button type="submit" name="btnLogout" id="btnLogout" class="btn-
enabled" formnovalidate>Kirjaudu ulos</button></div><br>
</form>
</body>
</html>
```

JSP-sivu: History.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<link rel="stylesheet" href="CommonCSS.css" type="text/css"/>
</head>
<body>
<div><h1>SRT (Self Register Tag)</h1></div>
<div><h2>Tapahtumahistoriasi</h2></div>
<hr>
<div><input type="text" class="input-text1" value="Käyttäjätunnus:"
readonly>
<input type="text" name="username" value="${username}" class="input-
text2" readonly></div><br>
<div><input type="text" class="input-text1" value="Etunimi:" readonly>
<input type="text" name="firstname" value="${firstname}" class="input-
text2" readonly></div><br>
<div><input type="text" class="input-text1" value="Sukunimi:"
readonly>
<input type="text" name="lastname" value="${lastname}" class="input-
text2" readonly></div><br>
<div><input type="text" class="input-text1" value="Opiskelijanumero:"
readonly>
<input type="text" name="idnumber" value="${idnumber}" class="input-
text2" readonly></div><br>
<hr>
<div><b>20 viimeisintä tapahtumaa, viimeisin ylimpänä:</b></div>
<div><textarea id="textHistory" rows="22" cols="75">${events}</tex-
tarea></div>
<hr>
<form action="History" method="post">
<div><button type=submit name="btnHistoryCancel" id="btnHistoryCancel"
class="btn-enabled">PERUUTA</button></div><br>
<hr>
<div><button type="submit" name="btnLogout" id="btnLogout" class="btn-
enabled">Kirjaudu ulos</button></div><br>
</form>
</body>
</html>
```

JSP-sivu: Logout.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<link rel="stylesheet" href="CommonCSS.css" type="text/css"/>
</head>
<body>
<div><h1>SRT (Self Register Tag)</h1></div>
<hr>
<br><div><b>Kiitos käytöstä. Poista sivuhistoria ja sulje se-
lain.</b></div><br>
<div><a href="Login.jsp"><b>Kirjaudu sisään.</b></a></div>
</body>
</html>
```

JSP-sivu: Timeout.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<link rel="stylesheet" href="CommonCSS.css" type="text/css"/>
</head>
<body>
<div><h1>SRT (Self Register Tag)</h1></div>
<hr>
<br><div><b>Istunto on vanhentunut.</b></div><br>
<div><a href="Login.jsp"><b>Kirjaudu sisään.</b></a></div>
</body>
</html>
```

JSP-sivu: Error.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<link rel="stylesheet" href="CommonCSS.css" type="text/css"/>
</head>
<body>
<div><h1>SRT (Self Register Tag)</h1></div>
<hr>
<br><div><b>Toiminto keskeytetty. Ei yhteyttä autentikointipalve-
luun.</b></div><br>
<div><b>Yritä hetken kuluttua uudelleen.</b></div><br>
<div><a href="Login.jsp"><b>Kirjaudu sisään.</b></a></div>
</body>
</html>
```

CSS-tiedosto: CommonCSS.css

```
@CHARSET "UTF-8";

div {
    margin: 0 auto;
    width: 400px;
    text-align: center;
}

.btn-login {
    -webkit-appearance: none ! important;
    width: 80px;
    height: 30px;
    padding: 2px;
    border-radius: 4px 4px;
    color: #212121;
    background-color: #90caf9;
    font-weight:bold;
}

.btn-enabled {
    -webkit-appearance: none ! important;
    width: 100px;
    height: 30px;
    padding: 2px;
    border-radius: 4px 4px;
    color: #212121;
    background-color: #00ff00;
    font-weight:bold;
}


.btn-disabled{
    -webkit-appearance: none ! important;
    cursor: not-allowed;
    pointer-events: none;
    width: 100px;
    height: 30px;
    padding: 2px;
    border-radius: 4px 4px;
    color: #bdbdbd;
    background-color: #eeeeee;
    font-weight:bold;
}

.input-text1{
  -webkit-appearance: none ! important;
  cursor: not-allowed;
  pointer-events: none;
  text-align: right;
  border: none;
  font-weight: bold;
  width: 190px;
}

.input-text2{
```

```
  -webkit-appearance: none ! important;
  cursor: not-allowed;
  pointer-events: none;
  text-align: left;
  font-weight: bold;
  width: 190px;
}

span{
    background:#F8F8F8;
    border: 5px solid #DFDFDF;
    color: #717171;
    font-size: 10px;
    height: 90px;
    letter-spacing: 1px;
    line-height: 13px;
    position: relative;
    text-align: left;
    top: -140px;
    left: -145px;
    display:none;
    padding: 0 20px;

}
span:after{
    content:'';
    position:absolute;
    bottom:-10px;
    width:10px;
    height:10px;
    border-bottom:5px solid #dfdfdf;
    border-right:5px solid #dfdfdf;
    background:#f8f8f8;
    left:50%;
    margin-left:-10px;
    -moz-transform:rotate(45deg);
    -webkit-transform:rotate(45deg);
    transform:rotate(45deg);
}
ps{
    margin:100px;
    float:left;
    position:relative;
    cursor:pointer;
}
ps:hover span{
    display:block;
}
```

Web.xml-tiedosto

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLoca-
tion="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/ja-
vaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
  <display-name>SRT02</display-name>
  <servlet>
    <servlet-name>Login</servlet-name>
    <servlet-class>com.srt02.pkg.Login</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>Login</servlet-name>
    <url-pattern>/SRT-JSP/Login</url-pattern>
  </servlet-mapping>
  <servlet>
    <servlet-name>Main</servlet-name>
    <servlet-class>com.srt02.pkg.Main</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>Main</servlet-name>
    <url-pattern>/SRT-JSP/Main</url-pattern>
  </servlet-mapping>
  <servlet>
    <servlet-name>PinAdd</servlet-name>
    <servlet-class>com.srt02.pkg.PinAdd</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>PinAdd</servlet-name>
    <url-pattern>/SRT-JSP/PinAdd</url-pattern>
  </servlet-mapping>
  <servlet>
    <servlet-name>PinRemove</servlet-name>
    <servlet-class>com.srt02.pkg.PinRemove</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>PinRemove</servlet-name>
    <url-pattern>/SRT-JSP/PinRemove</url-pattern>
  </servlet-mapping>
  <servlet>
    <servlet-name>TagAdd</servlet-name>
    <servlet-class>com.srt02.pkg.TagAdd</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>TagAdd</servlet-name>
    <url-pattern>/SRT-JSP/TagAdd</url-pattern>
  </servlet-mapping>
  <servlet>
    <servlet-name>TagRemove</servlet-name>
    <servlet-class>com.srt02.pkg.TagRemove</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>TagRemove</servlet-name>
    <url-pattern>/SRT-JSP/TagRemove</url-pattern>
  </servlet-mapping>
  <servlet>
```

```
    <servlet-name>History</servlet-name>
    <servlet-class>com.srt02.pkg.History</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>History</servlet-name>
    <url-pattern>/SRT-JSP/History</url-pattern>
  </servlet-mapping>
  <listener>
    <listener-class>com.srt02.pkg.CommonHttpSessionListener</listener-
class>
  </listener>
  <filter>
    <filter-name>CommonHttpCacheFilter</filter-name>
    <filter-class>com.srt02.pkg.CommonHttpCacheFilter</filter-class>
  </filter>
  <filter-mapping>
    <filter-name>CommonHttpCacheFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
  <session-config>
    <session-timeout>1</session-timeout>
  </session-config>
</web-app>
```